DOCUMENT RESUME

ED 180 765	SE 029 383
AUTHOR	Lew, Art
TITLE	Project SOUL: Computer Training Program for High School Students from Disadvantaged Areas. Part III, The Scientific Programming Course. Technical Report.
INSTITUTION	University of Southern California, Los Angeles. Dept. of Electrical Engineering.
SPONS AGENCY	National Science Foundation, Washington, D.C.
REPORT NO	USCEE-402-C
PUB DATE	Apr 71
GRANT	NSF-GJ-0981
NOTE	135p.
EDRS PRICE	MF01/PC06 Plus Postage.
DESCRIPTORS	Compensatory Education: *Computer Oriented Programs: Computers: *Computer Science Education: Curriculum Development: Disadvantaged Youth: *Educationally Disadvantaged: Mathematics: Program Content: *Program Descriptions: Program Evaluation: *Programing: Sciences: *Secondary Education: Summer Programs

ABSTRACT

This report details the Scientific Programming Course that is a part of "Project SOUL." The course is intended for underprivileged high school juniors and seniors having an interest in mathematics or science and aspirations to attend college. The report is divided into three sections. In section I, the administration and operation of the course as it was offered in the summer of 1970 is discussed. In section II, the contents of the course are described. This section may be used as a guide for the design of curricula for future courses of the same nature. Evaluation of the course and suggested improvements are given in section III of the report. Various course materials and information are appended. (MK)

****

*   Reproductions supplied by EDRS are the best that can be made   *
*   from the original document.   *

****

April 1971

USCEE 402C

U.S. DEPARTMENT OF HEALTH,
EDUCATION & WELFARE
NATIONAL INSTITUTE OF
EDUCATION

THIS DOCUMENT HAS BEEN REPRODUCED EXACTLY AS RECEIVED FROM THE PERSON OR ORGANIZATION ORIGINATING IT. POINTS OF VIEW OR OPINIONS STATED DO NOT NECESSARILY REPRESENT OFFICIAL NATIONAL INSTITUTE OF EDUCATION POSITION OR POLICY

"PERMISSION TO REPRODUCE THIS
MATERIAL HAS BEEN GRANTED BY
Mary L. Charles
NSF

TO THE EDUCATIONAL RESOURCES
INFORMATION CENTER (ERIC)."

ED180765

UNIVERSITY OF SOUTHERN CALIFORNIA

Technical Report

PROJECT SOUL:

Computer Training Program for High School Students from Disadvantaged Areas

PART III - The Scientific Programming Course

Art Lew

Supported by the National Science Foundation
under Grant No. GJ-0981

ELECTRONIC SCIENCES LABORATORY

Engineering

SE 029 383

PART III CONTENTS

THE SCIENTIFIC PROGRAMMING COURSE 1

1. Introduction 1

SECTION I 2

2. Admission and Placement 2
3. Organization 3
4. Personnel 4
5. Course Material 6

SECTION II 7

6. Course Objectives 7
7. Teaching Guide 7
8. Design Philosophy 30

SECTION III 32

9. Assessment 32
10. Recommendations 34

APPENDICES

A. Organizational Chart
B. Schedule of Classes
C. Aptitude Test
D. Supplementary Notes
E. Laboratory Problems
F. Additional Homework and Examinations
G. Samples of Student Work
H. Evaluation Questionnaire

THE SCIENTIFIC PROGRAMMING COURSE

1. Introduction

The Scientific Programming Course is part of "Project Soul," a computer training program for high school students from disadvantaged areas. The Scientific Programming portion of the project is intended for underprivileged high school juniors and seniors having an interest in mathematics or science and aspirations to attend college. It is designed to help close the competition gap that may confront them in the future by compensating for educational deficiencies and by providing a headstart. The program provides valuable background upon which these young persons can build in learning computer science, and will enable them to use the computer to solve problems in mathematics, engineering and science, from the earliest stages of their college careers. The immediately practical aspect, as well, is significant: the students receive training for computer-related jobs which can help them in financing their educations on a continuing basis.

This report on the Scientific Programming Course is divided into three sections. In Section I, we discuss the administration and operation of the course as it was offered in the summer of 1970. Further details and general project information may be found in the general report. In Section II, we describe the contents of the course. This section may be used as a guide for the design of curricula for future courses of the same nature. In Section III, we provide an evaluation of the course, and suggest improvements or modifications again for future courses. Various course materials and information are appended.

SECTION I

2. Admission and Placement

The Scientific Programming Course was designed for mathematics or science oriented students. The only scholastic requirement we had imposed, however, was minimal: a year of high school algebra with average or above grades. Our rationale for this is that programming aptitude is not sufficiently well-correlated with the usual academic subjects to warrant limitation to above average students. It is more important that the student have an interest in the subject and, of course, an aptitude for the peculiar rigors of programming. The interest is important, for without motivation little can be learned.

The aptitude for programming is reflected in the abilities to reason logically, to solve simple mathematical problems, and, especially to carry out tedious algorithms with precision. An aptitude test was designed to detect the foregoing abilities. (See Appendix C). It was used as part of the admission process, and also for class placement. A prime concern was that we attempt to minimize the chances of (relative) failure and its concomitant discouragement by excluding those students who in all likelihood would not profit from the course, and by keeping the level of students within each class fairly uniform. It was also decided to exclude students having more than 20 hours of previous course exposure to computer programming since they would not derive as much benefit as the untrained and would present certain problems in the classroom.

3. Organization

The Scientific Programming Course is an 80-hour course. It was offered in two consecutive sessions in order to keep class sizes relatively small while accommodating 150 students. The length of each session was four weeks, consisting of five days a week of lectures, laboratory work, and individual instruction as necessary, totaling four hours a day, half in the morning and half in the afternoon. The enrollment in each session of 75 students was divided into three classes of 25 students each. A schedule of classes is given in Appendix B.

The placement of students was based in large measure on the results of the aptitude test, which was also used as a guide for admission. The test was given three weeks prior to the start of the first session. Applicants were given the opportunity to express their preferences for sessions at that time. A course calendar is appended (Appendix A of Part I (The General Report)).

A typical day for the students was comprised of two hours of classroom instruction (lectures and discussion) and two hours of laboratory work, in either order. In the laboratory the students were provided with practical programming experience. There were daily assignments which kept each student busy coding, debugging, or keypunching. Laboratory assistants were present at all times to answer questions and to provide whatever other assistance the students required. Tutors were also available for private consultation.

4. Personnel

The personnel participating in the Scientific Programming Course is shown in the organization chart (Appendix A). The Principal Investigator, Project Director, and Technical Administrator had responsibility for the overall management of the Project. The Scientific Programming staff consisted of a Curriculum Coordinator, Instructors, Laboratory Assistants, Tutors, and special aides.

The responsibilities of the Curriculum Coordinator were to:

1. Develop the curriculum for the course.
2. Prepare a general teaching outline, including daily lesson plans, reading assignments, and laboratory problems.
3. Design a programming aptitude test to assist in the selection of students.
4. Conduct orientation and briefing sessions for the instructional staff.
5. Coordinate and supervise the work of the instructional staff, and assist in the implementation of the curriculum as the need arises.
6. Submit a detailed personal evaluation of the course in general and to give recommendations for future projects.

The responsibilities of the Instructors were to:

1. Present daily lectures according to the lesson plans provided. (Variations at the discretion of the Instructors were permitted.)
2. Assign reading, laboratory problems, and other homework.
3. Design, administer, and grade examinations.

4. Supervise the activities of the Laboratory Assistants.
5. Refer students to the Tutors for individual instruction as the need arises.
6. Monitor the progress and make a final evaluation of each student.

The responsibilities of the Laboratory Assistants were to:

1. Do the laboratory problems (before assignment to students).
2. Give students instructions on the use of the keypunch.
3. Check laboratory and other homework assignments.
4. Assist students in writing and debugging their programs, and in interpreting error messages.
5. Maintain accurate records of each student's progress in their laboratory work.
6. Recommend tutorial assistance.
7. Provide feedback to the Instructors.

The responsibilities of the Tutors were to:

1. Answer any questions posed by the students.
2. Provide individual tutoring to students as the need arises.
3. Provide feedback to the Instructors.

The special aides consisted of (i) job carriers, who were responsible for the submittal and return of student jobs from the computer center, and (ii) secretarial help, who typed, ran dittos, collated, and sundry other tasks.

5. Course Material

The text adopted for this course is Computer Science: FORTRAN Language Programming, co-authored by A. I. Forsythe, T. A. Keenan, E. I. Organick and W. Stenberg, which was published by John Wiley and Sons, Inc., New York, 1970. Its companion volume, Computer Science: A Primer, is used as a reference but not required. Supplementary notes written especially for the course include:

1. "An Introduction to Computers and Programming," by A. Lew.
2. "Basic Computer Concepts," by A. Lew.
3. "Numbers and Their Representations," by E. Angel and A. Lew.

These are appended. (See Appendix D).

Practical programming experience was provided by means of a series of laboratory assignments. (See Appendix E for a listing of the lab problems). These assignments required each student to write actual FORTRAN programs to be run. Keypunches were available for the students to punch their own programs onto cards. The programs were then run on an IBM System/360 Model 44 computer (housed in the Systems Simulation Laboratory of the University of Southern California School of Engineering).

SECTION II

6. Course Objectives

In developing the curriculum, our primary goal was this: upon successful completion of the course, the student should be able to write simple computer programs to solve whatever numerical problems he might encounter later in college or on the job. Our rule of thumb is that, if the student can solve a problem on paper, he should be able to solve it using a computer. To this end, the course provides training in the use of digital computers to solve problems in mathematics and science. Actual experience in programming an IBM System/360 computer using the FORTRAN IV language is given. Most importantly, general concepts basic to the understanding of any computer or programming language are discussed. Knowledge of FORTRAN and basic computer concepts should thus enable the student to readily learn other computer languages as the need arises. The preparation provided by this course should also accelerate the rate at which the student can learn to handle the more complex problems with which he is sure to be confronted.

7. Teaching Guide

In the following pages, we present three outlines of the Scientific Programming Course: (1) a topical outline, (2) a chronological outline, and (3) class schedules. The topical outline is a listing of the topics covered in the course. The topics are grouped under three headings: (I) Basic Concepts, (II) FORTRAN IV, and (III) Mathematical Topics. The chronological outline is listed by lessons. The subject matter for each of sixteen lessons is briefly noted. Four class schedules are then provided. Each schedule distributes some subset of the lessons over a four-week span.

Detailed lesson plans follow the schedules. For each lesson, we describe first the subject matter to be discussed in classroom lectures. We then suggest reading assignments which supplement the lectures. Finally, we provide laboratory problems to illustrate in a concrete way the concepts discussed and read, and to provide practical programming experience.

COURSE OUTLINE (TOPICAL)

I. BASIC CONCEPTS
- a. The stored-program concept
- b. Basic computer operations
- c. Compilers and operating systems
- d. Algorithms and flow charts
- e. Programs and languages
- f. Tracing a program
- g. Representation of numbers

II. FORTRAN IV
- a. FORTRAN elements
- b. Input and output
- c. Assignment statements
- d. Conditional branching
- e. Subscripting
- f. Iteration
- g. Subprograms

III. MATHEMATICAL TOPICS
- a. Number systems
- b. Roundoff
- c. Quadratic equations
- d. Euclidean algorithm
- e. Square root
- f. Sine function
- g. Roots of equations
- h. Averages and deviations
- i. Areas under curves
- j. Gaussian elimination

COURSE OUTLINE (CHRONOLOGICAL)

LESSON	SUBJECTS
1	A general introduction. Sample programs.
2	The stored-program concept. Basic computer operations. Compilers and operating systems.
3	Algorithms and flow charts. Tracing a program. FORTRAN as an algorithmic language.
4	FORTRAN elements. Input-output. Carriage control.
5	Number systems. Integer numbers. Real numbers. Computer storage of numbers. Roundoff.
6	Assignment statements and other FORTRAN elements. Integer and real representation of numbers. Real arithmetic.
7	Decisions, conditions, and branching. Logical expressions. Quadratic equations.
8	Literals and spacing. Euclidean algorithm. REVIEW.
9	Subscribed variables. Memory allocation. Input and output of arrays. Applications.
10	Iteration and DO loops. Input-output and nested DO loops. The iteration box.
11	Square root. Sine function. Associated mathematics.
12	Subprograms. Arguments. Examples.
13	Roots of equations. (The Newton-Raphson Method).
14	Averages and deviations. (Statistics).
15	Areas under curves. (Integration).
16	Gaussian elimination. (Matrix algebra).

COURSE OUTLINE (SCHEDULES)

DAY	SCHEDULE A	SCHEDULE B	SCHEDULE C	SCHEDULE D
1	Lesson 1	Lesson 1	Lesson 1	Lesson 1
2	Lesson 2	Lesson 2	Lesson 2	Lesson 2
3	Lesson 3	Lesson 3	Lesson 3	Lesson 3
4	Lesson 4	Lesson 4	*	*
5	Lesson 5	Lesson 5	Lesson 4	Lesson 4
6	Lesson 6	*	*	*
7	Lesson 7	Lesson 6	Lesson 5	Lesson 5
8	Lesson 8	Lesson 7	Lesson 6	*
9	Lesson 9	Lesson 8	*	Lesson 6
10	Lesson 10	*	Lesson 7	*
11	*	Lesson 9	Lesson 8	Lesson 7
12	Lesson 11	Lesson 10	*	*
13	Lesson 12	Lesson 11	Lesson 9	Lesson 8
14	Lesson 13	*	Lesson 10	*
15	Lesson 14	Lesson 12	*	Lesson 9
16	Lesson 15	Lesson 13	Lesson 11	*
17	Lesson 16	Lesson 14	Lesson 12	Lesson 10
18	*	*	*	*
19	FINAL	FINAL	FINAL	FINAL
20	Miscellany	Miscellany	Miscellany	Miscellany

(NOTE: * indicates discussion, review, and/or evaluation).

LESSON 1

SUBJECT MATTER

1. A general introduction:

Discuss "An Introduction to Computers and Programming."

2. Sample programs:

Describe the Payroll program (Fig. 1-7, p. 11 of text) and the Fibonacci program (Fig. 1-3, p. 10). The objective is to give a feel for what FORTRAN programs look like, and hence the description may be somewhat superficial. (Say so, lest you scare anyone). Pass around sample decks with control cards so that the student may see a typical job deck. Also pass out job listings.

READING ASSIGNMENT

1. "An Introduction to Computers and Programming"

LAB PROBLEM

1. Punch and run the Fibonacci program (Fig. 1-4) with the appropriate control cards. (This assignment is designed to introduce keypunching, card layout, deck set-up, and job submittal. The present goal is not understanding the program, but just the mechanics of running a job. In fact, the program as is has an error which should be ignored for now).

LESSON 2

SUBJECT MATTER

1. The stored-program concept:

Discuss the concept that, in addition to data, program instructions are stored and interpreted. Distinguish between compilation time and execution time. Referring to the Payroll program, explain the fact that input and output occur at execution time, whence input data is separate from the FORTRAN statements and outputted answers appear after the program is compiled (translated) and loaded.

2. Basic computer operations:

Discuss the basic capabilities (limitations) of computers. In particular, computers can do arithmetic, replace data at one place in its memory by that at another, read and write numbers and letters, and test the signs of numbers and take alternate actions accordingly. Illustrate how the Payroll and Fibonacci programs involve only those operations.

3. Compilers and operating systems:

Discuss, just to give a general idea, the functions of compilers and operating systems. Compilers translate programs written in a symbolic language (FORTRAN) into more basic computer operations. Operating systems allow jobs to be "batched" giving control alternatingly to translating programs and user programs.

READING ASSIGNMENT

1. "Basic Computer Concepts"

LAB PROBLEM

2. Punch and run the Payroll program (Fig. 1-7) with appropriate input data and control cards. (Make this just another keypunching exercise. Specify how data cards should be punched).

LESSON 3

SUBJECT MATTER

1. Algorithms and flow charts:

Discuss algorithms and flow charts. Use examples in the Primer. Refer also to Figures 1-2 and 1-6 of text. The basic flow chart boxes are (a) assignment, (b) decision, and (c) input and output. Note that these boxes correspond to the basic computer operations discussed in the previous lesson. The assignment box combines the arithmetic and replacement operations. The decision box involves the test and branch operation. The input and output boxes correspond to reading and writing. (A fourth box, the iteration box, is not discussed until Lesson 10).

2. Tracing a program:

Hand-trace the Fibonacci program and other examples. (Hand-tracing is very instructive because the student then in essence acts as a computer himself). (Note that a complete trace of the Fibonacci program is long enough to allow many students to help).

3. FORTRAN as an algorithmic language:

Compare the flow charts and FORTRAN programs in Fig. 1-2 and 1-6. Note the correspondences of the flow chart boxes to the FORTRAN statements.

READING ASSIGNMENT

1. Text, Ch. 1, pp. 3-11.

LAB PROBLEM

3. Determine what the following program computes by hand-tracing it.

```
          N =6
          K = 1
          I = 1
7         IF(I. GT. N) GOTO50
          K = K * I
          I = I  + 1
          GOTO7
50        WRITE  (6, 33)N, K, I
33        FORMAT (3I10)
          END
```

Verify your answer by punching and running the program. If it doesn't come out as expected, rerun it with the WRITE statement moved up one notch (i. e., interchanged with the GOTO7 statement).

LESSON 4

SUBJECT MATTER

1. FORTRAN elements:

Discuss (a) the FORTRAN character set, (b) statement labels, (c) variable names, (d) variable types, and (e) the INTEGER and REAL declarations. The I-N rule should be treated as an axiom, which must be memorized.

2. Input-output:

Discuss the READ, WRITE, and FORMAT statements. For now, just explain the I and F format codes. Emphasize the necessity for input and output lists to match formats (on a 1-1 basis). The equivalence of blanks to zeroes on input data cards should also be pointed out.

3. Carriage control:

Treat axiomatically the fact that the first character on every printed line acts as a carriage control character and is never printed. Introduction of leading blanks by using large field widths is simplest technique for obtaining single line spacing. (Use of H-formats, quote marks, and X-formats are discussed in Lesson 8).

READING ASSIGNMENT

1. Text, Section 2-1, pp. 12-15.
2. Text, Section 2-2, pp. 20-29.
3. Text, Section 2-4, pp. 42-44.

LAB PROBLEM

4. Rerun the Payroll program by
   (a) changing the FORMAT statements,
   (b) varying the order of the input and output lists,
   (c) experimenting with various data cards.

The objective is to learn by making perturbations to the program and observing the effects.

LESSON 5

SUBJECT MATTER

1. Number systems:

Discuss the binary, hexadecimal, and other radix systems, BCD numbers and conversion from one to another.

2. Integer numbers:

Discuss sign-magnitude, 1's and 2's complement, and excess-2^n representations of fixed-length binary integers.

3. Real numbers:

Discuss scientific notation, mantissas and exponents, E-format numbers, and finite precision.

4. Computer storage of numbers:

Discuss the ways in which numbers are stored in the IBM 360. (2's complement integers; base 16 with excess-64 exponent reals.)

5. Roundoff:

Discuss the truncation of numbers due to finite word length. Distinguish between "rationality" in base 10 and base 2.

READING ASSIGNMENT

1. "Numbers and Their Representations"
2. Text, Sections 5-1, 5-2, pp. 118-122.

LAB PROBLEM

5. Run the following program for the data of your choice (non-zero). Explain the results.

```
      READ (5,1)I, F
    1 FORMAT (F10.0, I10)
      WRITE (6,1) I, F
      WRITE (6,1) F, I
      SUM = I + F
      WRITE (6,1) SUM, SUM
      END
```

LESSON 6

SUBJECT MATTER

1. Assignment statements and other FORTRAN elements: Discuss (a) arithmetic expressions, (b) mixed-mode arithmetic, (c) operators and their precedence values, (d) predefined functions and function arguments, (e) length of statements and column 6, and (f) blanks in statements.

2. Integer and real representations of numbers: Discuss the change of representations that may occur by use of the assignment statement and by mixed-mode arithmetic. Describe integer division and the truncation function CHOP.

3. Real arithmetic: Illustrate the round-off errors that may accumulate in using real fixed-word-length arithmetic. Show the loss of significant digits when subtracting nearly equal numbers. Mention that the associative and distributive laws need not hold for computer arithmetic. (See Chapter 5 of the Primer). (Some of this material may be postponed until Lesson 11).

READING ASSIGNMENT

1. Text, Section 2-1, pp. 16-17.
2. Text, Sections 2-3, 2-4, pp. 30-41.

LAB PROBLEM

6. Write assignment statements for the expressions of Exercise 3, p. 18 of text. Punch and run them, assuming appropriate values for the variables.

LESSON 7

SUBJECT MATTER

1. Decisions, conditions, and branching:

Discuss the arithmetic IF, logical IF, and computed GOTO statements. Show variations on the Fibonacci program and others using the different statements.

2. Logical expressions:

Discuss relational operators, truth values, logical operators, and general logical expressions. Explain the extended precedence table (Table 3-2, p. 76).

3. Quadratic equations:

Flowchart a program to determine the two roots of a general quadratic equation

```
A * X ** 2 + B * X + C = 0
```

using the quadratic formula. Test for the special case where A = 0. Recall that the sign of the discriminant reveals the nature of the roots.

4. Mathematics:

The quadratic formula can be <u>derived</u> by the method of completing squares. Do so if time permits. The formula can be <u>verified</u> by plugging back into the equation.

READING ASSIGNMENT

1. Text, Section 2-1, pp. 18-20.
2. Text, Section 3-1, pp. 51-53.
3. Text, Section 3-4, 3-5, pp. 64-75.

LAB PROBLEM

7. Write and run a program to solve the general quadratic equation. Input the coefficients A, B, C and output the roots. Loop back to read in more data. Enough sets of data should be used to exercise all parts of the program.

LESSON 8

SUBJECT MATTER

1. Literals and spacing:

Discuss the use of H-fields and quote marks to output literal messages, and to control the printer carriage. Also discuss the X-field.

2. Mathematics:

Explain the two Euclidean algorithm programs (Figures 3-5 and 3-7). Consult the Primer. Trace the subtraction form for A = 12, B = 8. Trace the division form for A = 100, B = 64.

3. Review:

Use the Tallying program (Figure 3-1, p. 56) for review. Flow chart it, explain each statement, assume typical data, and trace.

READING ASSIGNMENT

1. Text, Sections 3-1, 3-2, 3-3, pp. 51-64.
2. Text, Section 4-1, pp. 103-105.

LAB PROBLEM

8. Add appropriate literal messages to Lab Problem 7. For example,

   a) for A = 0, write, "A = 0, single root = ..."
   b) for discriminant = 0, write, "double root = ..."
   c) for discriminant 0, write, "real roots are ..."
   d) for discriminant 0, write, "complex roots are ..."

Everyone should design his own formats.

LESSON 9

SUBJECT MATTER

1. Subscripted variables:

Discuss the use of single and double subscripts in arithmetic expressions and assignment statements (both left and right hand sides). Distinguish between XK and X_K. Illustrate, for example, the problem of averaging 10 numbers with and without subscripts. Use of subscripts will require an IF loop; trace it showing how subscripts change values, thereby addressing different words in memory.

2. Memory allocation:

Discuss the DIMENSION statement, computer storage of subscripted variables, and the column-wise storage of matrices. Note that lack of DIMENSION declarations may lead to ambiguities since we can no longer distinguish between array names and function names in arithmetic expressions.

3. Input and output of arrays:

Introduce the use of implied DO loops in read and write lists. (This is discussed further in the following lesson).

4. Applications:

Sample applications of singly-subscripted variables (vectors) include 1-D tables, functions of one independent variable (esp. time), coefficients of polynomials and other series. Applications of doubly-subscripted variables (matrices) include 2-D tables, functions of spatial (x - y) coordinates, coefficients of linear systems of equations.

READING ASSIGNMENT

1. Text, Sections 3-6, 3-7, pp. 75-86.

LAB PROBLEM

9. Write a program to read in a list of aptitude test scores (graded between 0 and 30), and to then determine the percentage that fall in the high (20-30), mid (10-19), and low (0-9) ranges.

LESSON 10

SUBJECT MATTER

1. Iteration and DO loops:

Discuss iterative procedures with and without fixed numbers of loops. Rewrite the Fibonacci and Payroll programs (of Chapter 1) using DO loops. (Both have no fixed loop limit). Discuss the problem of averaging N numbers using a DO loop. (This was done using an IF loop in the previous lesson). Illustrate how DO loops can always be replaced by IF loops.

2. Input-output and nested DO loops:

Continue the discussion on implied DO loops. Show how matrices may be inputted with and without implied DO loops. This will also require explanation of nested DO loops and of how FORMAT statements are refused.

3. The iteration box:

Describe the iteration flow chart box. Note that it is just a combination of two assignment statement boxes (for initialization and incrementation) and a decision box. Beware of the nonequivalence with DO loops described on pages 96 and 97 of the text. Rewrite Lab Problem 3 using a DO statement.

READING ASSIGNMENT

1. Text, Chapter 4, pp. 87-102, 106-117.

LAB PROBLEM

10. Write a program to
    a) read in a 5 x 5 matrix rowwise using implied DO loops,
    b) print it out rowwise without implied DO loops
    c) print it out columnwise without any DO loops (using IF statements),

d) print out the largest element in the matrix,

e) print out the sum of the principal diagonal elements.

These tasks should be done in separate parts of the same program, and appropriate literal messages should identify the various outputs.

LESSON 11

SUBJECT MATTER

1. Square root:

Discuss the use of Newton's method to determine square roots of positive reals. Newton's method is an iterative procedure (also called a method of successive approximations) in which, starting from an initial guess, successive iterations lead to improved answers. In other words, to solve a "fixed-point" problem of the form $X = G(X)$, the formula $X_{i+1} = G(X_i)$, $X_o=C$, is used. (Convergence of the procedure depends upon the nature of G, and usually also upon C.)

2. Sine function:

Discuss the use of a power series to approximate the function sin(X). Only a finite number of terms may be used, hence the series method is akin to polynomial approximation.

3. Mathematics:

Continue discussion of successive approximations, and of series expansions. Introduce the concept of convergence in both contexts. Distinguish between the errors arising from finite word lengths (rounding) and those arising from using a finite number of iterations or of terms (truncation).

READING ASSIGNMENT

1. Text, Sections 5-3, 5-4, 5-7, pp. 122-127.

LAB PROBLEM

11. Write a program to calculate π to six significant digits using the series (a) $\frac{\pi^2}{6} = \lim_{N\to\infty} \sum_{i=1}^{N} \frac{1}{i^2}$

and

(b) $\frac{\pi}{4} = 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \cdots$

LESSON 12

SUBJECT MATTER

1. Subprograms:

Discuss functions and subroutines. Explain the roles played by the CALL, FUNCTION, SUBROUTINE, RETURN, and END statements. Emphasize that subprograms are compiled independently of the programs that call them or which they call in turn.

2. Arguments:

Discuss function and subroutine arguments, their various types, and the restrictions imposed. Emphasize the one-to-one correspondence required. Mention alternate exits and function name arguments.

3. Examples:

Convert earlier programs (except for Lab Problem below) to subroutines or functions.

READING ASSIGNMENT

1. Text, Section 6-1, 6-2, 6-3, 6-4, 6-5, pp. 131-146.

LAB PROBLEM

12. Convert the quadratic equation program (Lab Problem 8) to a subroutine. Put all input-output statements in the main program.

LESSON 13

SUBJECT MATTER

1. Roots of equations:

Discuss the bisection method for finding roots (zeroes) of equations.

2. Mathematics:

Discuss the Newton-Raphson method for finding roots of equations. This will require introduction of the concept of derivatives. (Note that the Newton-Raphson method is a fixed-point method. Furthermore, the Newton method for finding square roots is a special case.) The calculus concepts of sequences, limits, and convergence will be required. Derivatives can be treated geometrically by discussing secants and tangents. In the Newton-Raphson formula, the derivatives can be approximated by using secants.

READING ASSIGNMENT

1. Text, Section 7-1, pp. 151-157.

LAB PROBLEM

13. Program Exercise 5, p. 157 of text, as a subroutine. Use this subroutine, instead of ZERO, to do Exercise 2, p. 157.

LESSON 14

SUBJECT MATTER

1. Averages and deviations:

Discuss averages (mean, median, mode) and deviations (variances, moments).

2. Statistics:

Discuss other statistics such as correlation, chi-square tests, etc.

READING ASSIGNMENT

1. Text, Sections 8-2, 8-3, pp. 169-171.

LAB PROBLEM

14. Extend Lab Problem 9 to determine the mean, median, mode, and standard deviation of aptitude test scores.

LESSON 15

SUBJECT MATTER

1. Areas under curves:

Discuss the rectangular, trapezoidal and Simpson formulas for finding areas under curves.

2. Mathematics:

Introduce the concept of integration. This will require the calculus concepts of sequences, limits, convergence.

READING ASSIGNMENT

1. Text, Sections 7-3, 7-4, pp. 161-165.

LAB PROBLEM

15. Determine π to six decimal places by finding the area under a quarter circle using Simpson's rule.

LESSON 16

SUBJECT MATTER

1. Gaussian elimination:

Discuss the Gaussian elimination method for solving systems of linear algebraic equations. Introduce the matrix form Ax = b. The elimination method essentially involves row operations on A.

2. Matrix algebra:

Discuss matrix arithmetic (addition, multiplication) and other matrix algebra topics (inversion, determinants).

READING ASSIGNMENT

1. Text, Section 8-1, pp. 166-169.

LAB PROBLEM

16. Program the Gauss elimination algorithm, as flowcharted in Figure 8-1, p. 167 of text. Add as many of the improvements described on p. 168 as possible.

8. Design Philosophy

In view of the objectives of the course, and of the backgrounds of the students, the Scientific Programming curriculum was designed with the following points in mind. First, the course should be taught as simply and straightforwardly as possible. There should be some redundancy to aid in the learning process. The students should not be required to acquire new information through reading. Reading assignments should be given on topics already covered by lectures, or else should be optional. Tutoring should be made readily available. In the belief that much can be learned by analyzing in depth sample programs, this tactic is relied upon. The student is also "led-by-the-hand" a great deal to facilitate transference of knowledge from sample programs to newly assigned ones.

The curriculum should not be rigidly structured, but rather should be flexible enough to adapt easily to any particular group of students. Consequently, the instructors should be allowed a great deal of freedom in the conduct of their classes. But while we cannot expect every class to proceed at the same pace, a certain minimal level of achievement should be set (Schedule D). This provides a definite target and assures that upon achievement, the student will have truly profited from the course.

Emphasis should be placed upon genuine understanding of basic and general concepts and not upon the details or idiosyncracies of specific languages. Consequently, the first portion of the course covers basic concepts, albeit in the FORTRAN context. While FORTRAN may eventually become obsolete, the underlying knowledge will remain of value. Thus no attempt should be made to teach as much as possible of FORTRAN IV at the expense of a weak treatment of a fundamental subset of the language. The subset stressed in the course is sufficient to program solutions to all problems the students are likely to encounter in the near future, however inefficiently or inelegantly. This is the reason four basic computer operations are emphasized, while other higher level operations (such as the iteration DO statement) are regarded as conveniences.

The main mathematical content of the course is left until the last week and is essentially optional material. The mathematics here and in earlier lessons should be presented at whatever level seems appropriate for any particular class. Since little mathematical maturity can be expected of the students, it is expected that the level will be relatively unsophisticated. This is not to say that new and computer-related mathematical topics should not be presented. In a scientific programming course, mathematics plays a very important role. But care should be taken that difficulties in fully understanding the mathematical aspects of a problem do not prevent students from successfully learning to program its solution, nor become a source of serious discouragement. When this is the case, the mathematics becomes expendable.

Treatment of numbers and their representations is of especial importance in understanding computer arithmetic. Regrettably, the mathematics involved is somewhat complicated. Very basic material on number systems and internal storage of numbers is included in this course, enough to distinguish between fixed and floating point representations of numbers, and to reveal the mysteries of roundoff. Full comprehension of these concepts is an unreasonable goal for this course, but knowledge of their existence and appreciation thereof is of value in itself.

Only a minimal number of laboratory problems were included in the lesson plans. These problems were designed to emphasize certain specific concepts, and were not meant to be comprehensive. The instructors were expected to supplement the problems with homework exercises and other laboratory problems of their own. They were also to design their own examinations. Copies of the contributions of the instructors appear in Appendix F.

SECTION III

9. Assessment

At its conclusion, the instructional staff was asked to evaluate the course. This was done by questionnaire, a copy of which is appended (Appendix H). We summarize here the main criticisms offered.

The concensus opinion was that in general the course was a fruitful and rewarding experience for staff and students alike. There was no doubt that non-honor high school students could be taught scientific programming in as little as four weeks. This was borne out again in the course this summer, as it had been the previous two summers in predecessor courses, as well as in courses across the country. A collection of programs (and their results) written by the students is included in Appendix G.

Most of the remarks made in last year's report remain valid. Academically oriented students, many of whom will use their newly acquired knowledge as a tool in their college work, became more enthusiastic about pursuing scientific research after having been exposed to the use of computers in solving problems. Less academically inclined students showed much more interest than they had in their high school science and math courses, which they claimed were often too theoretical and irrelevant to their needs. There were, unfortunately, a number of students who lacked either sufficient motivation or perseverance or aptitude, and consequently did not derive much benefit from the course. A major question is whether a more careful pre-evaluation and admission policy might have or should have eliminated these students from the start.

A question raised by one instructor was that of whether our students should be exactly those who have found no motivation in their traditional educational experience. In pre-session discussions, it was decided that there was very little we, having no formal educational or psychological training, could do with such students. In any case, it was concluded that the problems these students face are so deep as to be impervious to a limited four-week attack. Our course was seen as providing an opportunity for those who are motivated and have the aptitude (albeit not reflected in their performances in high school classes), rather than as attempting to provide motivation to those who lack same.

While it was our objective to provide preparatory rather than remedial training, we must still be concerned with the motivation of the students, at least to the extent of not discouraging anyone. This is a demanding requirement and necessitates a conscientious effort on the part of all the staff. The extent to which the staff was successful in this effort is difficult to measure.

Other criticisms made related to the content of the course and to the text. The text was not found to be very helpful. This did not, however, present a serious problem. The course had been designed with most of the burden placed on the instructors. Furthermore, programming more so than other subjects is learned by doing. Hence, although the text should be chosen wisely, it is more important to concentrate on improving the quality and efforts of the instructional staff. It should be noted that no book examined was found to be entirely suitable for this course.

The course content was generally found to be satisfactory. Some instructors expressed a preference for a different ordering of subjects and a slight shift in emphases. The DO statement was thought to be too closely linked with subscripted variables, and a simplification of input-output by use of standardized formats was suggested. Some students also had difficulty with various mathematical concepts, but a deemphasized exposure to the concepts is of sufficient value to make their complete elimination a dubious alternative.

Finally, the desire for the inclusion of more exercises and laboratory problems in the lesson plans was expressed. More could be easily found in the multitude of FORTRAN texts that abound, and consequently there was no real difficulty in leaving additional assignments to the discretions of the instructors. One other criticism was that the laboratory problems provided were too easy at the start while relatively difficult and time-consuming near the end of the course. One suggestion for remedying this is discussed in the concluding section.

10. Recommendations

We conclude this report on the Scientific Programming Course with a list of recommendations for improving the operation and content of the course. We first should emphasize that the course has been found to be extremely valuable and should be offered again, perhaps on an annual basis. Furthermore, we recommend that the course, or a facsimile, be offered across the country for there is need everywhere. Recommendations for future courses include the following.

A. ADMISSION. The more effort made in pre-evaluation of applicants, the more successful the course is likely to be. There is of course a practical limit to the amount of screening that can be performed, but some attempt should be made to limit enrollment to those showing motivation and aptitude. (We discuss this further below.)

B. SCHEDULE. Because of the nature of the subject, it is recommended that the course be taught with a shifting schedule. For example, during the first week(s) of the class, three-fourths of the day should be lecture, one-fourth laboratory; during the final week(s), this would be reversed, with one-fourth lecture and three-fourths laboratory. The middle week(s) may be split evenly between lecture and laboratory.

C. STAFF. Great care should be taken in the recruitment of the instructional staff. Desirable qualities for instructors, laboratory assistants, and tutors include knowledgeableness, enthusiasm, and a sincere desire to help. Of course, some prior experience in teaching would be preferable, but this has been found to be unnecessary. Of greater importance is the ability to interact well.

D. INTERACTION. It is of great importance to establish rapport with each student. This has the benefits of increasing the motivation of the students, aiding thereby in their learning process, and as a consequence rewarding the members of the instructional staff, stimulating thereby their enthusiasm, whence also their effectiveness. The instructors should see each student individually as often as feasible and should encourage class discussion by all. The laboratory assistants should take the initiative in soliciting questions from the students, for those who need the most help generally will not seek it. It is recommended that this be done by reviewing each student's progress periodically each day in his presence. This would encourage the students to make better use of their time in the laboratory, and would benefit the more reserved or shy students who would otherwise remain withdrawn. The instructors and laboratory assistants should not hesitate to refer students to the tutors; again this should not be voluntary, but should also not be an ordeal nor a stigma. The tutors, for their part, should be well versed in the Socratic method for best effectiveness. Ideally, the students should regard each member of the instructional staff as a friend to whom they may readily and naturally turn.

E. CONTENT. The content of the course is basically sound. However, for future courses with different emphases, some modifications would be appropriate. A more concrete job-oriented course would result by eliminating most of the basic introductory material and starting with FORTRAN elements. (FORTRAN texts by and large are of this nature.) A less mathematical course would result by deleting Lesson 5 and treating the other mathematical topics in a cookbook fashion. A course for honor students would result by discussing each subject more deeply, providing more motivation and background for the various concepts, and also by strengthening the mathematical content. A simplified course would result by eliminating most of the discussion of input-output problems and utilizing standardized formats.

F. DURATION. While the progress made by most students in the eighty hours of the course was most gratifying, there is little doubt that a longer course would be of great benefit. EDP school programming courses, for example, provide five to ten times the hours of instruction. College courses, on the other hand, generally provide fewer hours (around 40 to 50) of instruction. (In college, however, laboratory work is not usually included in the above hour count as the students are expected to do most of their work on their own outside the classroom.) In any case, it is recommended that the duration of the course be increased.

G. OBJECTIVES. The limitation to motivated and apt students is justifiable in relation to our stated objectives and capabilities. Regrettably, many students, and in particular those who have a substantial need, cannot meet our admission criteria. This leads us to the question of whether or not our objectives are too narrow, and whether a similar course should be designed for those who have the aptitude but lack motivation or for those who have the motivation but lack the aptitude. It should be emphasized that a course for these neglected students must necessarily differ in content from the present course. Furthermore, to handle such students would require a well-trained (in educational theory and psychology) instructional staff. It is felt that efforts along these lines would be of avail and value, and therefore it is recommended that such remedial courses be designed and implemented.

H. SCOPE. Because of the success of the program this year, and of the programs of the previous two years, we feel that the Scientific Programming Course has proven its value. Therefore, we again recommend an expansion of the program to accommodate as many underprivileged youths as funds, facilities, and personnel will allow. Furthermore, we recommend that similar programs be made available across the country, wherever there are students, who because of inadequate schools and a lack of community concern find themselves increasingly at a disadvantage. We note that programming courses are presently made easily available to students at middle or higher-class schools, and a multitude of courses for honor students abound. There is no sound reason for neglecting the average but capable high school youth, who may not have been as fortunate as others, but who would take advantage and benefit from whatever opportunities he may at long last encounter.

APPENDIX A

Organizational Chart

"PROJECT SOUL" 1970

Principal Investigator:
Richard Bellman

Project Director:
Carlos Ford-Livene

Technical Administrator:
M. Virginia Zoitl

Scientific Programming Curriculum
Coordinator: Art Lew

Session I:
June 29-July 24

Instructors:

1. Andy Yakush
2. Dan Tuey
3. M. Wasserman

Lab. Assistants:

1. A. Leon
2. P. Kumar
3. G. Nagel
4. R. Schein

Tutor:

1. C. Shoemaker

Session II:
July 27-August 21

Instructors:

1. Andy Yakush
2. Dan Tuey
3. B. Kashef

Lab. Assistants:

1. A. Leon
2. P. Kumar
3. M. Wasserman
4. R. Schein

Tutor:

1. G. Bloom

APPENDIX B

Schedule of Classes

COURSE	DATE	CLASS SIZE	TIME	LOCATION	INSTRUCTOR	KEYPUNCH DISTRIBUTION
Scientific Programming, Section A	June 29-July 24	25	10-12 Lab. 1-3 Lect.	OHE 210 VHE 310	A. Yakush	OHE 210-5kps
Scientific Programming, Section B	June 29-July 24	25	10-12 Lect. 1-3 Lab.	VHE 310 OHE 210	D. Tuey	OHE 210-5kps
Scientific Programming, Section C	June 29-July 24	25	10-12 Lect. 1-3 Lab.	VHE 314 OHE 212	M. Wasserman	OHE 212-5kps
Scientific Programming, Section D	July 27-August 21	25	10-12 Lab. 1-3 Lect.	OHE 210 VHE 310	A. Yakush	OHE 210-5kps
Scientific Programming Section E	July 27-August 21	25	10-12 Lect. 1-3 Lab.	VHE 310 OHE 210	D. Tuey	OHE 210-5kps
Scientific Programming Section F	July 27-August 21	25	10-12 Lect. 1-3 Lab.	VHE 314 OHE 212	B. Kashef	OHE 212-5kps

APPENDIX C

Aptitude Test

INSTRUCTIONS FOR PART I

In Part I you will be given some problems like those on this page. The letters in each series follow a certain rule. For each series of letters you are to find the correct rule and complete the series. One of the letters at the right side of the page is the correct answer. Look at the example below.

		(1)	(2)	(3)	(4)	(5)
W.	a b a b a b a b	a	b	c	d	e

For this problem, the series goes: ab ab ab ab
The next letter in the series is a. Choice 1 is the correct answer.

		(1)	(2)	(3)	(4)	(5)
X.	a a b b c c d d	a	b	c	d	e

In Example X above, the series goes like this: aa bb cc dd. The next letter in the series is e. Choice 5 is the correct answer.

Now do Example Y below.

		(1)	(2)	(3)	(4)	(5)
Y.	c a d a e a f a	d	e	f	g	h

In Example Y, the series goes: ca da ea fa. Therefore, the correct answer is g, Choice 4.

Now do Example Z.

		(1)	(2)	(3)	(4)	(5)
Z.	a x b y a x b y a x b	a	b	c	x	y

In Example Z, the series goes like this: axby axby axb. Therefore, the correct answer is y, Choice 5.

In the problems on the following page, you are to select the correct letter on the right-hand side of the page which belongs next in the series. Indicate the correct answer on the answer sheet.

PART I

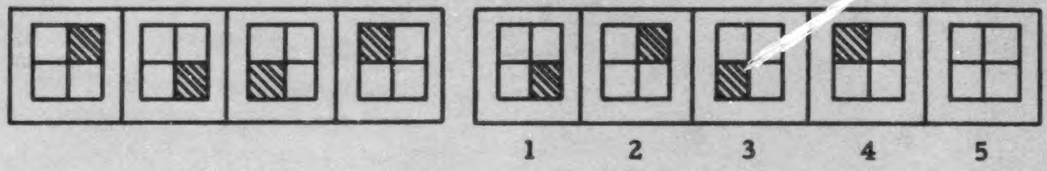
		(1)	(2)	(3)	(4)	(5)
1.	c d e x y z f g h x y z	i	j	k	l	m
2.	t s r t s r t s	r	s	t	v	w
3.	a b c c d e f f g	e	f	g	h	i
4.	m n m n k l o p o p k l	k	o	p	q	r
5.	a b c i j d e f i j	g	h	i	j	k
6.	a i b c i d e f	e	f	g	h	i
7.	a g b h c	d	f	g	h	i
8.	a e d h g	h	i	j	k	l
9.	k s j t i u h	v	w	x	y	z
10.	n j f m i e l	d	h	i	j	m

INSTRUCTIONS FOR PART II

In Part II you will be given some problems like those on this page. Each row is a problem. Each row consists of four figures on the left-hand side of the page and five figures on the right-hand side of the page. The four figures on the left make a series. You are to find out which one of the figures on the right-hand side would be the next or the fifth one in the series. Now look at Example X.

Example

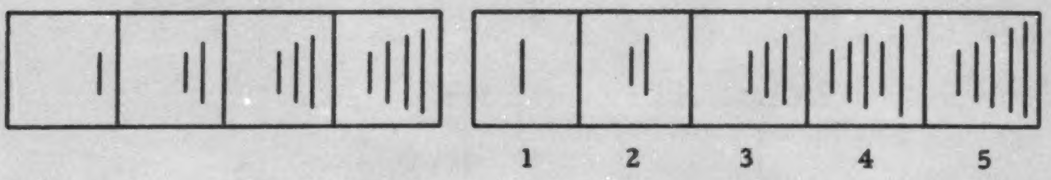
X



In Example X there is a clockwise movement of the striped square: upper right, lower right, lower left, upper left. The next or fifth position in this clockwise movement would thus be upper right, and so Choice 2 is the correct answer.

Now look at Example Y.

Y

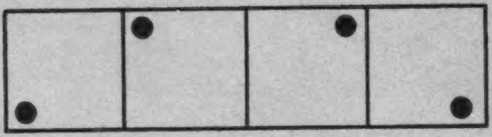


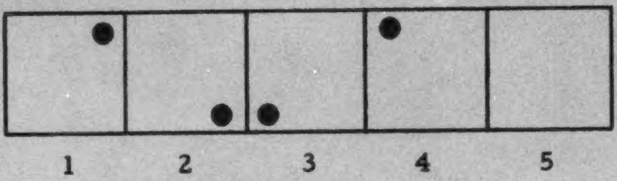
In the series of figures on the left, there is one more line in each box and these lines increase in length. Now look at the five choices on the right-hand side of the page and determine the correct answer.

You should have selected Choice 5 which has five lines, one more than the last box on the left with the fifth line slightly longer than the last line in Box 4.

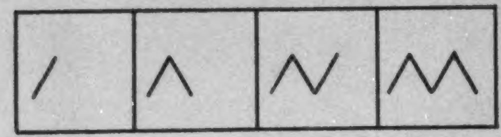
PART II

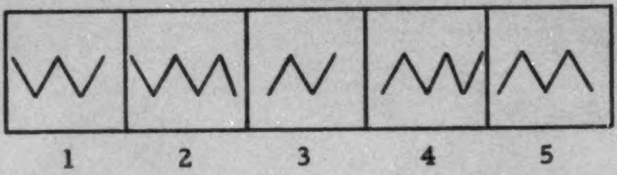
11.



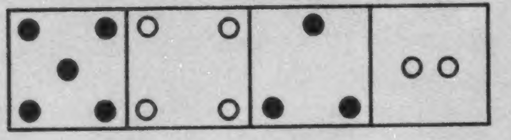


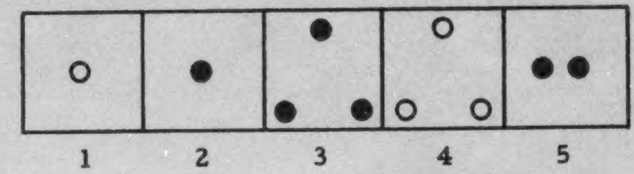
12.



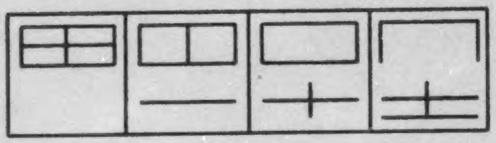


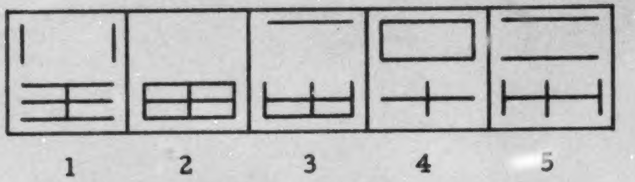
13.



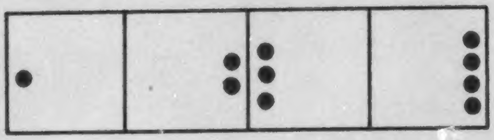


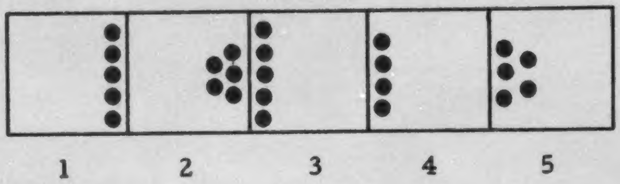
14.



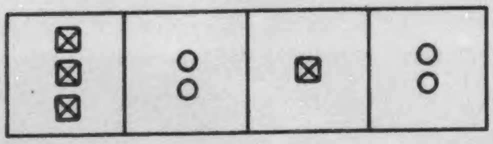


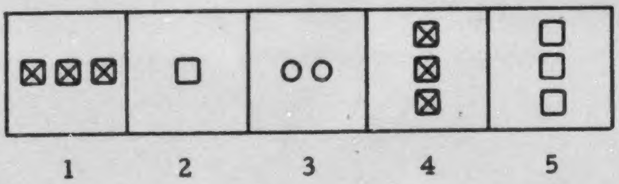
15.



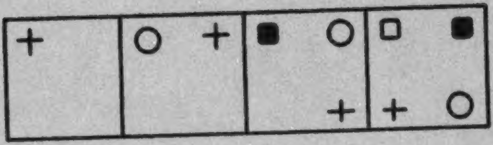


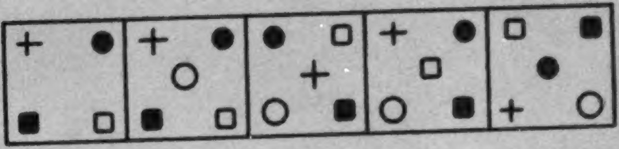
16.





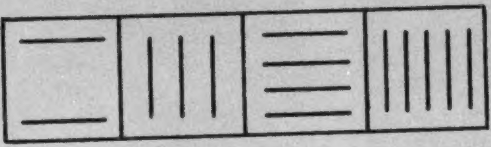
17.

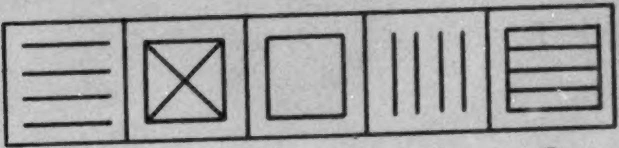




1 2 3 4 5

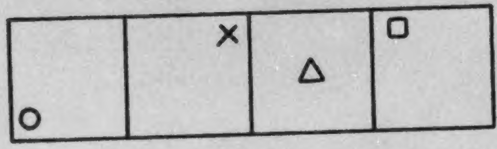
18.

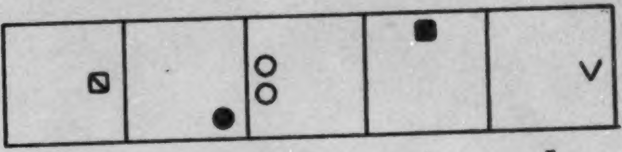




1 2 3 4 5

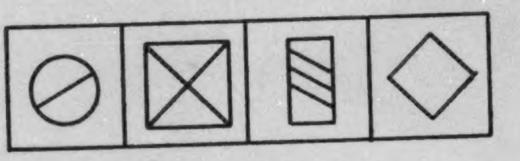
19.

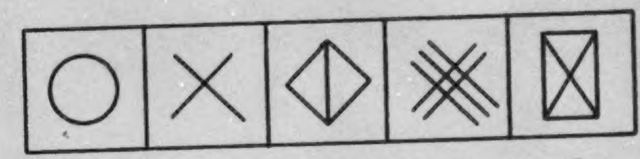




1 2 3 4 5

20.





1 2 3 4 5

INSTRUCTIONS FOR PART III

In Part III you will be given some problems in arithmetical reasoning. After each problem there are five answers, but only one of them is the correct answer. You are to solve each problem and indicate the correct answer on the answer sheet. The following problems have been done correctly. Study them carefully.

Example X: How many apples can you buy for 80 cents at the rate of 3 for 10 cents?

(1) 6 (2) 12 (3) 18 (4) 24 (5) 30

The correct answer to the problem is 24, which is Choice (4).

Example Y: In 4 weeks John has saved $2.80. What have his average weekly savings been?

(1) 35¢ (2) 40¢ (3) 50¢ (4) 70¢ (5) 80¢

The correct answer to the above problem is 70¢, Choice (4).

PART III

21. A clerk multiplied a number by ten when it should have been divided by ten. The answer he got was 100. What should the answer have been?

(a) 1 (b) 10 (c) 100
(d) 1000 (e) 10,000

22. The average salary of three programmers is $95 per week. If one programmer earns $115, a second earns $65, how much is the salary of the third programmer?

(a) $95 (b) $105 (c) $115
(d) $160 (e) $180

23. If a card punch operator can process 80 cards in half an hour, how many cards can she process in a seven and one-half hour day?

(a) 560 (b) 600 (c) 800
(d) 1120 (e) 1200

24. In a programming team of 12 persons, 1/3 are women and 2/3 are men. To obtain a team with 20% women, how many men should be hired?

(a) 4 (b) 6 (c) 8
(d) 12 (e) 20

25. It cost a college 70 cents a copy to produce the program for the homecoming football game. If $15,000 was received for advertisements in the program, how many copies at 50 cents a copy must be sold to make a profit of $8000?

(a) 14,000 (b) 35,000 (c) 46,000
(d) 75,000 (e) 115,000

PART III

26. Express the number 1921.02 in scientific notation.

(a) 19.2102×10^2 (b) 1.92102×10^3

(c) $.192102 \times 10^4$ (d) $192102. \times 10^{-2}$ (e) 1921.02

27. If $f(x) = \dfrac{3x - 1}{2x^2 - 3x - 2}$, then $f(1/3)$ is:

(a) 1 (b) 1/3 (c) 0

(d) $-\frac{25}{9}$ (e) ∞

28. Solve for x, given that $3x + 1 = 7$

(a) 7 (b) $\frac{4}{3}$ (c) 3

(d) 2 (e) none of these

29. Solve the system of equations $x + y = 12$
$x - y = 4$

(a) $x = 8, y = 4$ (b) $x = 16, y = 8$ (c) $x = 9, y = 3$

(d) $x = 12, y = 8$ (e) none of these

30. Solve for x, given that $|3x - 4| \leq 7$.

(a) $-1 \leq x \leq 11$ (b) $\frac{1}{3} \leq x \leq \frac{11}{3}$ (c) $-1 \leq x \leq \frac{11}{3}$

(d) $-11 < x < 3$ (e) none of these

APPENDIX D

Supplementary Notes

AN INTRODUCTION TO COMPUTERS AND PROGRAMMING

1. Introduction

Many people regard digital computers with awe* because of the miraculous powers often attributed to them. This is due in part to the fact that computer designers and users have tended to be more anthropomorphic than necessary. One speaks of a computer's "memory," and of its "reading" and "writing" abilities, for example. It is no wonder then that computers are sometimes popularly referred to as "electronic brains." We shall see, however, that whatever computers can do today, or for generations to come, is entirely dependent on the brain power of the human programmer. Our immediate objective shall be the elimination of the mysticism that pervades the air regarding what a computer really is and what it can do.

2. It's Really Just An Adding Machine

A digital computer is more aptly likened to an ordinary adding machine than to a human brain. It is no secret that a computer does arithmetic and does it well. It adds, subtracts, multiplies, and divides--all at extraordinary speeds and with incredible accuracy. So it should come as no surprise that computers and adding machines have this characteristic in common: an internal mechanism, which we'll call an "arithmetic unit," by which numbers can be operated upon. (By "operated," we mean added, subtracted, etc.)

Our own experience with adding machines--a cash register is a good example of a particular one--tells us that somewhere inside is another unit by which numbers can be stored. We know this since subtotals can be remembered by the machine. Computers share what may be termed a "storage unit."

---

*This is not to say that awe is totally unwarranted. But awe based on ignorance rather than knowledge should be reserved for the aesthetic and the divine.

We also know that some means must exist for controlling the internal operation of an adding machine. Depending on which button is pushed, the adding machine can be "instructed" to add or subtract, to display a subtotal or total, to clear itself, and so forth. The mechanism which regulates the actions of the adding machine is called the "control unit," a unit also shared by computers.

A further analogy may be made in that adding machines and computers both have "input/output units"--that is, mechanisms by which numbers may be entered and "instructions" given, and by which the results of the operations upon the given numbers may be displayed. Adding machines have buttons on a keyboard for input, and have "windows" and/or printed paper tapes for output. Computers have a larger variety of input/output media. In our everyday lives, we have all encountered punched cards which may be used for both input and output. Printed records, or "listings," are also common for output.

We thus arrive at a model, pictured in Figure 1, for both adding machines and computers. The presence of the control unit at the hub is more figurative than factual. It is meant only to represent whatever controls the internal operation and interactions of the other units.

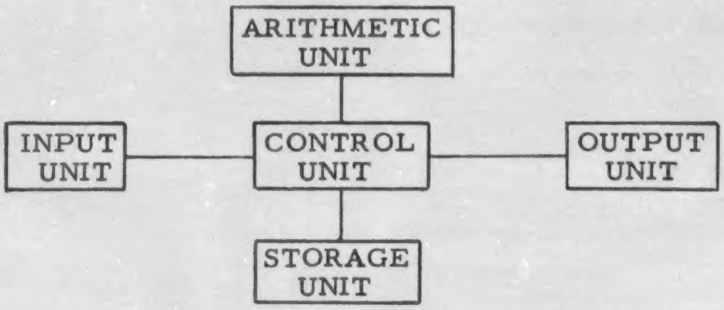


Figure 1

We see now that adding machines and computers may, in one sense at least, be regarded as basically the same. This statement, however, should be interpreted with care, for there does exist a fundamental, even crucial, difference--that of the "stored-program" concept. Before elaborating on this, we will first consider how we might instruct a person who had access to an adding machine to calculate the answers to a simple problem. We do so because, in principle, the approach to be taken is directly analogous to the approach we must take when using a computer.

3. What Was That Again?

Suppose we wished to calculate the average of a collection of numbers, as must be done, for example, to determine the average score of a class after an exam. An averaging procedure for five numbers may be described as follows: add the numbers and divide by five. It may seem that this description is unambiguous, but that is because we have all taken averages before. To a person who does not know what an average is, and who perhaps has never added more than two numbers at a time, the above description would undoubtedly be inadequate. A more explicit averaging procedure for five numbers is as follows:

Step 1. Take the first number.
Step 2. Add the second number to the result of Step 1.
Step 3. Add the third number to the result of Step 2.
Step 4. Add the fourth number to the result of Step 3.
Step 5. Add the fifth number to the result of Step 4.
Step 6. Divide the result of Step 5 by the number five.

The above set of instructions is still somewhat incomplete. For if we were to assign the present task to an idiot--who could do simple arithmetic but was otherwise devoid of intellectual faculties (and it would not be improper to think of a computer as such an idiot)--then it is likely that we must be more specific. For example, we must tell the idiot precisely which number is first, second, third, and so forth, lest he be confused as to the order in which to add the numbers (although, in this example, it is irrelevant), and further we must tell him that the answer we seek is the result of the last step (Step 6) and to report it.

This example was contrived to emphasize the fact that, when dealing with computers, we must take great pains to be unambiguous. For we cannot assume that our intent, however obvious to us, can be communicated without stating it explicitly, completely, and precisely.

4. A New Language, As If We Didn't Have Enough Trouble With English

We now note two other inadequacies in the foregoing six-step procedure. First, a more concise means of representation should be sought so that we can state instructions for averaging 100, or 1000, or more numbers in a more compact form. Second, a more general means of representation should be sought so that we can use the same set of instructions to obtain the averages of 100, and 1000, and more, numbers. To average an "array" of N numbers, $X = \{X_1, X_2, \ldots, X_N\}$, we might reformulate our instructions as follows:

Step 1. Let our subtotal have the value zero.

Step 2. Repeat Step 3 letting I take each of the integer values from 1 up to N inclusive.

Step 3. Add the I-th number (of the array X) to the "current" value of the subtotal, and then let the subtotal have that sum as its new value.

Step 4. Divide the "current" value of the subtotal by N, and then let the subtotal have that quotient as its new value.

The word "current" in the above is meant to imply that the steps are performed sequentially in time, and thus that the subtotal changes in value as the steps are executed. We emphasize that Step 1 is executed first, then Step 2 causes Step 3 to be executed N times (where I changes each time Step 3 is executed), and finally Step 4 is executed. Thus the subtotal changes N + 2 times.

The above is rather wordy, and hence we adopt the following shorthand language:

```
1  SUBTOTAL=0
2  DO 3 I=1,N
3  SUBTOTAL=SUBTOTAL+X(I)
4  SUBTOTAL=SUBTOTAL/N
```

We shall call a set of instructions written in this shorthand language a "program." A close comparison of this program with the more verbose version will reveal how this new language is to be interpreted. But a few explanatory notes should be made.

First observe that the values of N and of the array X are assumed to be known to whoever (or whatever) uses the program. Stated in another way, N and X are variables, whose values depend on the particular set of numbers we wish to average, and thus must be specifically defined for each new set. To average the three numbers 5, 10 and 15, a complete program would therefore be as follows:

```
    N=3
    X(1)=5
    X(2)=10
    X(3)=15
    SUBTOTAL=0
    DO 3 I=1,N
3   SUBTOTAL=SUBTOTAL+X(I)
    SUBTOTAL=SUBTOTAL/N
```

We observe from the above that it is really unnecessary to number each line or "statement," since we can by convention assume that the statements are to be executed sequentially in the order given. Not having to write statements numbers for each line of the program makes life easier for the typist. Also, for typographical reasons, X_I has been rewritten X(I).

A final observation is that the equal signs used in the program signify replacement rather than equivalence. In other words, in the statement numbered 3, we cannot cancel SUBTOTAL and conclude that X(I)=0 as we would in algebra. The correct interpretation is that the expression on the right-hand side of the equal sign is to be evaluated and this value is then to replace the previous value of the variable named on the left-hand side. We emphasize again that the instructions of the program are to be executed one at a time, independently of the other instructions, in a sequential manner. In other words, computers basically operate by sequentially replacing the values associated with variables in a manner dictated by programs written in a language such as the above.

5. Surprise!

The shorthand language used above is FORTRAN.

BASIC COMPUTER CONCEPTS

In our previous lesson, we have seen what a typical computer program looks like. In particular, a FORTRAN program consists of a set of statements which are instructions to a computer somehow specifying the manner in which values of variables are to be sequentially determined. We shall attempt to explain here how a computer actually goes about executing such a program.

It should first be emphasized that computer languages such as FORTRAN cannot be understood directly by a computer. One reason for this is that, whereas we have an extended alphabet of 26 letters, 10 digits, and a dozen or so special characters or punctuation marks at our disposal in writing a computer program, computers on the other hand only have two digits (zero and one) in its alphabet. Therefore each character in our extended alphabet must have a distinctly coded internal form. (In the IBM 360, for example, an A is represented internally by the string of ones and zeros 10100001.) The encoding process from external to internal form is done automatically, so far as we are concerned, via the magic of electronics.

When a computer reads in a FORTRAN program, it will have stored in its memory a very long string of ones and zeros. What occurs then is a translation process of some complexity. We can only hope to describe here in very simple terms the essential features of the translation process. The underlying concept is that the translator (called a "compiler") is in reality someone else's previously translated program, whose function is to read in programs in our external language, encoded automatically as a string of ones and zeros, and to then translate or convert this encoded string (which we emphasize is numeric) into a different numeric string which the computer can then understand. The translated string (which we call the "object code") is said to be in a "machine language."

In order to obtain a better understanding of the nature of machine language, we shall first discuss what basic operations a computer can actually perform. Its repertoire is surprisingly small.

1. A computer can perform the four basic arithmetic operations--addition, subtraction, multiplication, and division.
2. A computer can sense whether a number is positive, zero, or negative, and can then alter its normal sequential operation depending upon what it senses.
3. A computer can perform input and output (read or write) operations, which involves direct and automatic encoding and decoding of information (from external to internal form or vice versa).
4. A computer can transfer numbers around in its memory.

A computer can do nothing else that does not involve these fundamental capabilities. Thus the task of the programmer is to devise schemes (algorithms) to solve complex problems utilizing only these basic operations.

Machine language consists of encoded instructions that specify which of the above operations is to be performed. The translation (compilation) process involves the conversion of FORTRAN instructions into machine language instructions. For example, the FORTRAN program

```
    SUBTOTAL = 0
    DO 8 I = 1 , N
8   SUBTOTAL = SUBTOTAL + X ( I )
    SUBTOTAL = SUBTOTAL / N
```

would be translated into machine language instructions that specify the following sequence of fundamental operations.

1. Transfer zero to SUBTOTAL.
2. Transfer 1 to I.
3. Add to SUBTOTAL the I-th element of X and transfer to SUBTOTAL the sum.
4. Add 1 to I and transfer to I the sum.
5. Subtract N from I (without changing N or I) and if the result is not positive, go back to Step 3.
6. Divide SUBTOTAL by N and transfer to SUBTOTAL the quotient.

Since the DO or iteration statement is not one of the fundamental operations, it must be converted, and as we can see it is equivalent to the incrementation, test, and branch operations specified in lines 4 and 5. The six instructions above indeed involve only the fundamental capabilities of a computer.

We have implicitly assumed that the symbols in our program (SUBTOTAL, I, N, and X) are associated with physical locations in the computer which are somehow distinguishable and which can be individually referenced. In fact, a computer's memory consists of a set of (usually several thousand at least) such memory locations, each of which we call a "word." To distinguish among different words, there corresponds to each a unique number, called its "address." (In analogy, we distinguish between houses by their street addresses.) Every number or variable in a program must be assigned to a particular location in memory, and every reference to that number or variable can then be made using its numeric address. It is the additional task of the compiler to produce machine language object code having such numeric references. The first instruction in the above program would then be, for example, "transfer the number at address 1000 to address 2000" supposing, of course, that the constant zero is at memory 1000 and that the variable SUBTOTAL has been associated with memory location 2000.

We have stated earlier that object code itself is a numeric string (of ones and zeros), and consequently it can be stored in the same memory locations as numbers and variables. It is a very fundamental fact that program instructions and program data can share the memory of a computer. (Of course, at any one instant, they must occupy separate words.) Looking at the number in an arbitrary word of a computer, it cannot be determined out of context whether that number is an encoded instruction or is simply program data. What then is a computer to do?

All is not lost. For computers have what is called a "location counter," which contains the address of the next instruction to be performed by the computer. We say that "control" is given next to the instruction specified by the location counter. This location counter is automatically (electronically) incremented by one instruction as each instruction is executed, except of course when the present instruction causes a transfer of control as after a sign test. For a correct program, then, instructions are distinguishable from data in that the location counter cannot have as its value the address of a word of data.

In the above, we have been discussing what is known as the "stored-program" concept. We conclude by describing the general operation of a computer as it translates and executes FORTRAN programs.

The location counter of the computer is initially set, magically if you will, to the first instruction of the compiler (which we assume is in object code form somewhere in the computer's memory).

The compiler instructions causes a FORTRAN program (which we will call the user program) to be read into the computer memory, and treating this program as numeric data, the compiler converts it to machine language. To execute the user program, the compiler can simply place the address of the first instruction of the user program into the location counter. The above are called the compilation and execution stages, respectively. So that it may regain control, the compiler at compile time adds to the end of the user program instructions to place in the location counter the address of the compiler. The compiler can then proceed to another user program when the first completes its job. This process is known as a batched operating system.

NUMBERS AND THEIR REPRESENTATIONS

1. Integers and Reals

The digital computer is designed to handle two distinct types of numbers. It is important to understand the difference between the two types since they are stored and operated on in entirely different manners. The first type of number is the integer or the fixed-point number. These numbers are the ordinary whole numbers, both positive and negative, such as

1,2,-5,0,12345,-79.

The other type of numbers are called the reals or floating-point numbers and are distinguished by the appearance of a decimal point in their representations (thus the names fixed and floating point). Examples of real numbers are

1.1, .234, 1234.56, -.0016.

Since an integer such as 2 can also be written as the real 2., we will only include the decimal point when we want to use "2" as a real. This point will become clearer when we discuss how the two types of numbers are stored and how they are operated on by the computer.

2. Scientific Notation

The real numbers which arise in the problems in which we are interested, may vary greatly in magnitude. In a single problem we might have numbers as large as 1,000,000,000,000 and others as small as .000 000 000 1. Scientific notation gives us a convenient way of writing these numbers. The first number can be written as 1.0×10^{12} and the second as 1.0×10^{-10}. We could also write the two numbers as $.1 \times 10^{13}$ and $.1 \times 10^{-9}$. For reasons which will become clearer later we will always use the second form. In general a number in scientific notation has the form

$$.m \times 10^{e}$$

where m is called the mantissa and e the exponent.

All real numbers are stored in the computer in scientific notation, that is by their mantissa and exponent. There are two reasons for this. The first is to conserve storage. Suppose we would like our computer, which we may assume for now is based on the decimal system, to handle numbers as big as 10^{99} and others as small as 10^{-99}. We would also like to be able to retain 10 digits of any number. If we were to try to store our numbers in the form

$$d_n d_{n-1} \cdots d_2 d_1 d_0 . d_{-1} \cdots d_{-k}$$

we would have to make provision in our computer for over 200 digits plus a sign. However, if we agree to store numbers by means of their mantissas and suitable powers of 10, we need only store 12 digits and two signs; 10 digits for the mantissa, 2 digits for the exponent, the sign of the mantissa and the sign of the exponent. This is a considerable saving of computer storage.

The second reason for storing reals by mantissa and exponent is the ease with which we can do multiplication and division. Using the same example as above, we see that if we were to do multiplication by ordinary methods, the arithmetic unit of our computer would have to be capable of multiplying two 100 digit numbers. This process is expensive in both time and equipment. On the other hand if we write our two numbers as $.m_1 \times 10^{e_1}$ and $.m_2 \times 10^{e_2}$, we know

$$(.m_1 \times 10^{e_1})(.m_2 \times 10^{e_2}) = .(m_1 m_2) \times 10^{e_1+e_2}.$$

We have accomplished the desired multiplication by multiplying two 10 digit numbers and adding two 2-digit numbers, a considerable saving of effort. Arithmetic operations involving integers are handled in the ordinary way. A typical digital computer has two distinct ways of storing numbers and two distinct ways of doing arithmetic.

3. Binary Numbers

We usually write a real decimal (or base 10) number in the form

$$d_n d_{n-1} \ldots d_2 d_1 d_0 . d_{-1} d_{-2} \ldots d_{-m}$$

where we understand the above number to be the sum

$$d_n \times 10^n + d_{n-1} \times 10^{n-1} + \ldots + d_1 \times 10^1 + d_0 \times 10^0 + d_{-1} \times 10^{-1} + \ldots + d_{-m} \times 10^{-m}.$$

The digits, d_i, can be chosen from the set $\{0,1,2,3,4,5,6,7,8,9 = 10 - 1\}$. We can extend the above ideas to get a representation for numbers in some other base, say r.

In general a number in base r can also be written as

$$d_n d_{n-1} \ldots d_1 d_0 . d_{-1} \ldots d_{-m}$$

where now this number is understood to be the sum

$$d_n \times r^n + d_{n-1} \times r^{n-1} + \ldots + d_1 \times r^1 + d_0 \times r^0 + d_{-1} \times r^{-1} + \ldots + d_{-m} \times r^{-m}$$

and the d_i are restricted to the set $\{0, 1, \ldots, r-1\}$. Since the digital computer is actually a binary machine, we should be interested in base 2 numbers. Each binary digit (or *bit*) can be 0 or 1 only. A typical binary number looks like 1011.1. To find the decimal equivalent of 1011.1, we write

$$1011.1 = 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 + 1 \times 2^{-1}$$

$$= 8 + 2 + 1 + \frac{1}{2} = 11.5_{10}.$$

The usual rules for addition, subtraction, multiplication and division apply to binary numbers. Below are some examples.

Binary	Decimal Equivalent
$\begin{array}{r} 101110 \\ +100011 \\ \hline 1010001 \end{array}$	$\begin{array}{r} 46 \\ +35 \\ \hline 81 \end{array}$
$\begin{array}{r} 101110 \\ -100011 \\ \hline 1011 \end{array}$	$\begin{array}{r} 46 \\ -35 \\ \hline 11 \end{array}$
$\begin{array}{r} 1001 \\ \times 101 \\ \hline 1001 \\ 0000 \\ 1001 \\ \hline 101101 \end{array}$	$\begin{array}{r} 9 \\ \times 5 \\ \hline 45 \end{array}$
$\begin{array}{r} 101 \\ 1001 \overline{)101101} \\ \underline{1001} \\ 1001 \end{array}$	$\begin{array}{r} 5 \\ 9 \overline{)45} \end{array}$

When we read a decimal number into the computer, the number is immediately converted into binary and stored. There are a number of methods for accomplishing this conversion. We will give an example of one such rule. Suppose we wish to convert the integer 125 to binary. Then we have

$$
\begin{aligned}
125 &= 62 \times 2 + 1 \\
62 &= 31 \times 2 + 0 \\
31 &= 15 \times 2 + 1 \\
15 &= 7 \times 2 + 1 \\
7 &= 3 \times 2 + 1 \\
3 &= 1 \times 2 + 1 \\
1 &= 0 \times 2 + 1.
\end{aligned}
$$

Taking the column of remainders bottom-up, we find

$$125_{10} = 11\ 11\ 101_2$$

The conversion from binary to decimal can easily be accomplished by having a table of values of 2^n available.

4. Hexadecimal

When numbers are written in binary form, the resulting strings of ones and zeros can become very cumbersome. For instance, the binary number

1001110101100001

cannot easily be recognized. For this reason hexadecimal or base 16 numbers are used as a shorthand for binary numbers. If

$$d_n d_{n-1} \ldots d_0 . d_{-1} \ldots d_{-m}$$

is a hexadecimal number then each d_i must be chosen from the set $\{0,1,2,\ldots,14,15 = 16-1\}$. However, if we are not careful, we could confuse the single hexadecimal character 12 with the two consecutive hexadecimal characters 1 and 2. To avoid this confusion, we replace the hexadecimal characters 10, 11, 12, 13, 14, 15 by A, B, C, D, E, F. Thus the hexadecimal number

$$12A.1_{16}$$

means

$$1 \times 16^2 + 2 \times 16^1 + 10 \times 16^0 + 1 \times 16^{-1}$$

$$= 256 + 32 + 10 + \frac{1}{16} = 298.0625_{10}.$$

It is very simple to convert binary numbers to hexadecimal and vice versa. Since $16 = 2^4$, each consecutive four bits of a binary number corresponds to a single hexadecimal digit. We merely have to use the following table

Decimal	Binary	Hexadecimal
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

Thus the binary number

```
1001   1101   0110   0001
  9      D      6      1
```

is $9D61_{16}$. Arithmetic with hexadecimal numbers follows directly from what we know of binary and decimal arithmetic. For example

$$\begin{array}{r} \text{A23B} \\ \underline{+\text{10C8}} \\ \text{B303} \end{array} \qquad \begin{array}{r} \text{D063} \\ \underline{-\text{50AA}} \\ \text{7FB9} \end{array}$$

It is important to become somewhat familiar with hexadecimal since most of the system 360 literature is written in terms of it.

5. Roundoff Errors

We all know that the definition of a rational number is: "One which can be written as the ratio of two integers." Numbers such as $\sqrt{2}$, $\sqrt{3}$, π, e are irrational. Mathematically, the distinction is very important but it is not very important as far as the computer is concerned. We can best demonstrate why this is so with an example. The number 1/3 is rational. However, since the computer can only recognize the number 1/3 as the result of dividing 1.0 by 3.0 or

$$.3333333333$$

we know that the above number will be stored as $.m \times 10^e$. Since our computer has only a finite memory we cannot possibly store the infinite number of 3's. The computer will chop off the number after some point and save perhaps only seven of the m. Thus, the number stored instead of 1/3 is .3333333 which is almost 1/3 but not quite. The error made due to the computer's ability to handle only finite length mantissas is called round-off error. Although each round-off error may be very small, these errors can accumulate and cause difficulties.

We will say that a number is rational in base r if it can be written exactly as

$$d_n d_{n-1} \ldots d_0 . d_{-1} \ldots d_{-m}$$

where each d_i is chosen from $\{0, 1, \ldots, r-1\}$ and m is finite. Thus while 1/3 is rational, it is irrational in base 10. The importance of this concept is that if we have a computer which operates in base r and the representation of a given number is both rational in base r and the mantissa length of the computer is greater than m + n + 1, the number will be stored correctly. For instance the binary number

10011 . 1101

will be stored correctly in a binary computer with mantissa length 9 but will be rounded off in a binary computer with mantissa length 8.

A number can be rational in one number system but irrational in another. For example, the decimal number .3 is rational base 10 but irrational base 2. Thus we cannot possibly store .3 exactly in any binary computer.

6. Computer Storage of Numbers

Although we know that our computer actually operates in binary, let us assume for now that we have a decimal computer. We will look at how real and integer numbers are stored. Later we will be able to make the necessary modifications to binary.

We will assume that our decimal computer has memory words 11 digits long. Every number we read into the memory will be allotted exactly 11 digits of storage. Associated with each set of 11 digits is a number, the address of the word. We will assume that the memory has 4400 digits or equivalently 400 words numbered from 000 to 399.

We will see how reals and integers are stored differently although each type is allotted only one word of storage. To avoid confusion we will write the number two as 2 when we want it stored as an integer but as 2. when we want it stored as real. In fact, this is exactly the method the computer uses to make this distinction.

7. Integer Storage (Sign-Magnitude)

Since we want to allow for both positive and negative integers we must use one digit, of the eleven we are allowed or our number, to store the sign. We will use a 1 in the first digit of a memory word to denote a minus sign and 0 will denote a plus sign. The rest of the 11 digits will be used to store the magnitude of the integer. Suppose we were to store the integer - 53 in memory address 44. Then if we look at memory word 44 we would find

1	0	0	0	0	0	0	0	0	5	3

while if we stored + 53 we would find

0	0	0	0	0	0	0	0	0	5	3

Since 10 digits are allowed for the magnitude of our integers we can store any integer in the range

$$-999,999,999 \text{ to } +999,999,999$$

or

$$-(10^{10} - 1) \text{ to } 10^{10} - 1.$$

If we were to try to read in an integer less than $-10^{10} + 1$ or greater than $10^{10} - 1$, the computer would print out an overflow message.

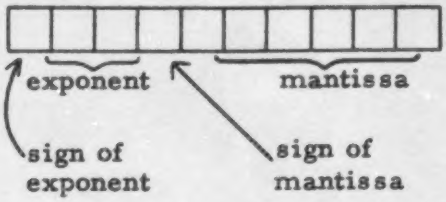
8. Real Storage (Mantissa-Exponent)

You should know by now that the real number 12.34 can also be written as $1234. \times 10^{-2}$ or $.1234 \times 10^{2}$ or even as

$123400000. \times 10^{-7}$. In practice any of these forms can be read into the computer correctly. However, no matter which form is read into the computer, the number is always stored as $.1234 \times 10^{2}$. That is, the computer always puts the decimal point to left of the most significant figure.

Since only one word of 11 digits is allocated to a real, part of the memory word will contain the exponent and part the mantissa. We will assume that 3 digits are allowed for the exponent and 8 for the mantissa. Since the exponent can be positive or negative, the first of the three bits will be used for the sign. The remaining two digits can then contain magnitudes of exponents ranging from 0 to 99. Thus if we tried to read in the number $.1234 \times 10^{100}$ we would get an overflow message.

The mantissa can also be positive or negative so the first of the eight bits allowed for the mantissa will contain a sign. The remaining seven digits will contain the seven most significant places of the mantissa. Thus a real word is stored as



Some examples are

$9.87654321 = .987654321 \times 10^{1}$

0	0	1	0	9	8	7	6	5	4	3

$-9.87654321 = -.987654321 \times 10^{1}$

0	0	1	1	9	8	7	6	5	4	3

$.0000000000987654321 = .987654321 \times 10^{-10}$

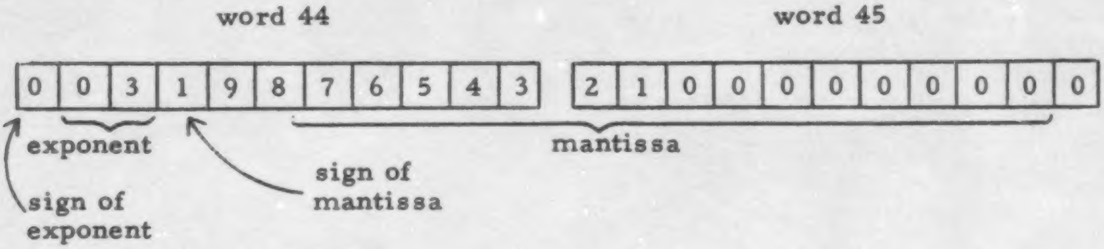
1	1	0	0	9	8	7	6	5	4	3

$-.00987654321 = -.987654321 \times 10^{-2}$

1	0	2	1	9	8	7	6	5	4	3

9. Double Precision

In some applications we would like to have more than seven significant figures for real numbers. We are allowed, at our option, to use two adjacent storage words to contain a single real number. Exponents are still restricted to lie between −99 and +99, therefore the additional 11 digits can all be used to store the extended mantissa. We now have 18 significant places. Thus if we store the number −987.654321 in double precision starting with word 44, the memory will contain



word 44

0	0	3	1	9	8	7	6	5	4	3

word 45

2	1	0	0	0	0	0	0	0	0	0

10. A Further Example

We have seen that the computer can store numbers in three distinct manners; as integers, as reals, or as double precision reals. As long as the numbers are entered properly, they will be stored correctly. However, if we were to look at a storage location, we could not tell what number the location contained unless we also knew what type of number was contained in the location. For example, suppose we looked at memory words 44 and 45 and they contained the following data word 44

1	0	1	1	3	4	1	0	0	2	3

word 45

0	0	0	1	1	2	3	0	0	0	0

The following interpretations are all possible:

1) 44 contains the integer −113410023
   45 contains the integer 11230000

2) 44 contains the real −.03410023
   45 contains the real −.123

3) 44 contains the integer .113410023
   45 contains the real −.123

4) 44 contains the real −.03410023
   45 contains the integer 11230000

5) 44 and 45 contain the double precision real
   −.034100230001123.

11. The IBM 360 Computer

We have discussed in the preceding sections the basic ideas behind computer representation of numbers. As you might expect, the actual representation of numbers in a real computer differs slightly from the earlier idealized version. We will now make the necessary corrections since we shall be dealing with a particular computer (an IBM 360). The major difference are that (a) the 360 operates in binary or hexadecimal, (b) integers are represented in two's-complement, and (c) reals are represented in the hexadecimal system with the exponent in excess-64. We shall elaborate upon these concepts in what follows.

The fundamental storage unit of the 360 is called a byte. Each byte consists of eight binary storage locations or bits (short for binary digits). The memory of our 360 computer consists of

16^4 (=65,536) bytes which are numbered in hexadecimal form from 0000 to FFFF. Each integer and real is allocated four bytes (=32 bits) of storage. Each such group of four bytes is called a word. Hence, our computer has 16,384 words of memory, each of which can contain an integer or a real (or possibly a machine language instruction). The allocation of memory to the various types of data is performed automatically by the compiler.

12. Representation of Integers in the 360

Integers are stored in a 360 word in binary two's-complement form. What this means is the following. Positive integers are stored in sign-magnitude form, the first bit of the word being the sign bit (0 for plus), and the remaining 31 bits containing the binary representation of the magnitude of the integer. Thus, the integer +100 would be stored as

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 1 0 0

The largest positive integer that can be stored in 31 bits is $2^{31} - 1$. Negative integers are not stored in sign-magnitude form, i.e., by just changing the sign bit of the representation of its complement. (The complement of a number N is defined as just −N. This definition holds whether N is positive or negative.) Negative integers are stored in two's-complement form, which is derived from its complement as follows: Invert each bit (i.e., change 0 to 1 and 1 to 0) in the representation of the complement, and then add the integer one. Thus, the integer −100 would be stored as

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 1 1 0 0

(Using binary arithmetic, the two's-complement of an integer N for a 32-bit word can also be defined as $2^{32} - N$. Note the difference between two's-complement and complement.)

13. Representation of Reals in the 360

Real numbers are translated before storage in the 360 from base 10 to base 16, i.e., to the form $.m \times 16^e$, where m (the mantissa) is hexadecimal and e (the exponent) is binary. The mantissa is stored in sign-magnitude form: the sign of the mantissa is contained in the first bit of the 360 word, with the magnitude being stored in the last three bytes of the word. Thus, we have 24-bit precision, which corresponds to approximately six or seven decimal place accuracy. The exponent is stored in the remaining seven bits (of the first byte) of the word in excess-64 form.

Excess-64 form means that to the binary value of the number e, the constant 64 is added prior to storage. For example, the exponent 0 is therefore stored as the seven-bit integer 1000000. The largest exponent which can be represented is +63 (1111111 in excess-64); the smallest exponent which can be represented is −64 (0000000 in excess-64).

Consider the 360 representation of the real number $-100._{10}$. Since

$$-100._{10} = -64._{16} = -.64_{16} \times 16^2 = -.01100100_2 \times 16^2$$

and the excess-64 form of the exponent 2 is $64_{10} = 1000010_2$, we arrive at the internal form

1 1 0 0 0 0 1 0 0 1 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

which is the 360 representation of the real −100. The above 32-bit binary string can be written in hexadecimal shorthand as A2640000.

Double precision reals use an additional word (four bytes) for extending the mantissa. The seven byte (=56 bit) mantissa yields approximately 15 decimal place accuracy.

APPENDIX E

Laboratory Problems

1. Punch and run the Fibonacci program (Figure 1-4 of text) with the appropriate control cards. (The program as is has an error which should be ignored for now.)

2. Punch and run the Payroll program (Figure 1-7) with appropriate input data and control cards.

3. Determine what the following program computes by hand-tracing it.

```
      N = 6
      K = 1
      I = 1
 7    IF ( I .GT. N ) GOTO 50
      K = K * I
      I = I + 1
      GOTO 7
 50   WRITE ( 6, 33 ) N,K,I
 33   FORMAT (3I10)
      END
```

   Verify your answer by punching and running the program. If it doesn't come out as expected, rerun it with the WRITE statement moved up one notch (i.e., interchanged with the GOTO statement).

4. Rerun the Payroll program

   (a) changing the FORMAT statements,
   (b) varying the order of the input and output lists,
   (c) experimenting with various data cards.

   The objective is to learn by making perturbations to the program and observing the effects.

5. Run the following program for the data of your choice (nonzero).

```
      READ(5,1)I,F
 1    FORMAT (F10.0,I10)
      WRITE(6,1)I,F
      WRITE(6,1)F,I
      SUM = I + F
      WRITE(6,1)SUM,SUM
      END
```

   Explain the results.

6. Write assignment statements for the expressions of Exercise 3, page 18 of text. Punch and run them, assuming appropriate values for the variables.

7. Write and run a program to solve the general quadratic equation, $Ax^2 + Bx + C = 0$. Input the coefficients A, B, C, and output the roots. Loop back to read in more data. Enough sets of data should be used to exercise all parts of the program.

8. Add appropriate literal messages to Lab Problem 7. For example,

   (a) for $A = 0$, write "A = 0, single root = ..."
   (b) for discriminant = 0, write "double root = ..."
   (c) for discriminant > 0, write "real roots are ..."
   (d) for discriminant < 0, write "complex roots are ..."

   Design your own formats.

9. Write a program to read in a list of aptitude test scores (graded between 0 and 30), and to then determine the percentage that fall in the high (20-30), mid (10-19), and low (0-9) ranges.

10. Write a program to

    (a) read in a 5×5 matrix rowwise using implied DO loops
    (b) print it out rowwise without using implied DO loops
    (c) print it out columnwise without using any DO loops (using IFs)
    (d) print out the largest element in the matrix
    (e) print out the sum of the principal diagonal elements.

    These tasks should be done in separate parts of the same program, and appropriate literal messages should identify the various outputs.

11. Write a program to calculate π to six significant digits using the series

    (a) $\pi = [(1 + 1/4 + 1/9 + 1/16 + \ldots) \times 6]^{\frac{1}{2}}$
    (b) $\pi = (1 - 1/3 + 1/5 - 1/7 + \ldots) \times 4$

12. Convert the quadratic equation program (Lab Problem 8) to a subroutine. Put all input-output statements in the main program.

13. Program Exercise 5, p. 157 of text, as a subroutine. Use this subroutine, instead of ZERO, to do Exercise 2, p. 157.

14. Extend Lab Problem 9 to determine the mean, median, mode, and standard deviation of the aptitude scores.

15. Determine π to six decimal places by finding the area under a quarter circle using Simpson's rule.

16. Program the Gauss elimination algorithm, as flowcharted in Figure 8-1, p. 167 of text. Add as many of the improvements described on p. 168 as possible.

APPENDIX F

Additional Homework and Examinations

FORMAT EXERCISE

```
1 0 4 8 0 1 5 0 1 1 0 1 5 3 6 0 2 0 1 1 8 1 6 4 7 9 1 6 4 6 6 9 1 7 9
```

I. Using the above card representation, perform the following input statements:

```
      READ(5,11)IN, I, J, N, K
11    FORMAT(I6, I2, I3, 2I2)
      IN=____, I=____, J=____, N=____, K=____

      READ(5,12)J, K, I, N, NUM, MOM
12    FORMAT(3(I3, I2))
      J=____, K=____, I=____, N=____, NUM=____, MOM=____

      READ(5,13)JOB, KOUNT, IST, LOVE, NEVER, MORE
13    FORMAT(2(2I2, I3))
      JOB=_______, KOUNT=_______, IST=_______
      LOVE=_______, NEVER=_______, MORE=_______
```

II. If the variables have the following values, perform the indicated output statements:

I=77921, J=6907, KARY=11008, LOVE=42751,
NOT=9100, MORE=21, IBLE=8, JOY=462,
KAST=93, LATE=48663, MONEY=22178, NOISE=593.

```
      WRITE(6,14)I, J, KAST, NOISE, NOT, MONEY
14    FORMAT(2I6, 2I4, 2I7)
```

```
      WRITE(6,15)LOVE, MORE, IBLE, LATE, KAST, IBLE
15    FORMAT(2(I3,I6), 2I6)
```

```
    WRITE(6,16)NOISE, JOY, MORE, KAST, I, MONEY
16  FORMAT(2I5/2I4/I8,I6)
```

III. Using the card representation in I., perform the following input statements:

```
    READ(5,17)ABLE,BAKER, CANDY, DOG, ENOCH, FUN
17  FORMAT(F6.1, F3.2, F8.4, 2F6.2)
```

ABLE=_______, BAKER=_______, CANDY=_______,

DOG=_______, ENOCH=_______, FUN=_______

```
    READ(5,18)PUPPY, QUIT, RUN, SAVE, TIME, ZEBRA
18  FORMAT(F8.3, 2F6.2, 3F8.1)
```

PUPPY=_______, QUIT=_______, RUN=_______,

SAVE=_______, TIME=_______, ZEBRA=_______

```
    READ(5,19)KOUNT, UN, KAIL, VOM, IT, IOP, BOTTOM,
        NEVER, TOP, LOVE
19  FORMAT(2(I3, 2(F3.2, I6))
```

KOUNT=____, UN=____, KAIL=____, VOM=____,

IT=____, IOP=____, BOTTOM=____, NEVER=____,

TOP=____, LOVE=____

```
815.25-7229.504.839-964232.87.88265 11
```

IV. Using the above card representation, perform the following input statements:

```
    READ(5,20)HAPPY, PUP, QUIT, RUN, STAY, TICK
20  FORMAT(F6.3, F5.2, F3.1, F5.2, F4.3, F6.4)
```

HAPPY=_______, PUP=_______, QUIT=_______,

RUN=_______, STAY=_______, TICK=_______

```
      READ(5,21)ZEBRA, YES, XX, WANT, VERY, UNDER
21    FORMAT(F4.2, F4.1, F6.1, F5.3, F7.2, F6.2)
```

ZEBRA=_______, YES=_______, XX=_______,

WANT=_______, VERY=_______, UNDER=_______

```
      READ(5,22)A, B, C, D, E, F
22    FORMAT(F2.1, F4.3, F3.2, F5.1, F4.2, F7.3)
```

A=_______, B=_______, C=_______,

D=_______, E=_______, F=_______

V. If the variables have the following values, perform the following output statements:

ABLE=90.725, BAKER=-522.10, CANDY=8397.2,
DOG=65831.0, ENOCH=5.3172, FAREL=.41961,
ZEBRA=323.88, YELLOW=57.375, XTRA=-7321.1,
WHITE=16.815, VENUS=2.1438, UP=1640.8

```
      WRITE(6,23)ABLE, YELLOW, CANDY, WHITE, ENOCH, UP
23    FORMAT(F6.2, F7.3, F7.1, F6.2, 2F8.2)
```

```
      WRITE(6,24)ZEBRA, BAKER, XTRA, DOG, VENUS, FAREL
24    FORMAT(F6.1, F9.3, F6.2, F10.2, F5.2, F5.3)
```

```
      WRITE(6,25)ZEBRA, YELLOW, WHITE, VENUS, ABLE,
         ENOCH, FAREL
25    FORMAT(2(F5.2, F6.11), 2F7.3/F8.5)
```

VI. **Write the following Expressions in FORTRAN:**

$\frac{s^2 - a^2}{(s^2 + a^2)^2}$ ______________________________

$\frac{s - a}{(s - a)^2 + b^2}$ ______________________________

$\frac{8a^3 s^2}{(s^2 + a^2)^3}$ ______________________________

$\frac{b^2 - \sqrt{a}}{(s - e^a)(b + \quad s)}$ ______________________________

LAB EXERCISES - SECTIONS IA and IID

1. Punch and run the Fibonacci program.

2. Punch and run the Elementary Payroll program.

3. No. 3 of original assignment sheet.

4. No. 5 of original assignment sheet.

5. Exercise 1-4, page 11 of text. In addition find the total amount paid and the average amount paid.

6. Exercise 2-2, page 25 of text.

7. Program the Quadratic equations program.

8. Add names to No. 5 for practice with Alpha format.

9. Postal program. Given the three measurements of a package to be mailed, find the largest dimension and the perimeter of the package the other way. Add these two numbers together. If the total is less than or equal to 72 inches, it may be mailed; if it is greater than 72 inches, it may not be mailed. Write a program to do the above writing out the three measurements, the final measurement, and whether or not it may be mailed. Use appropriate literal messages.

10. Given data cards with names and a test score on each, determine if the score is low, medium or high, and write out the name of the person, his score, and his rating. Also find the highest score, the average score, the lowest score, and what percentage got each rating.

11. Some program for practice with nested DO loops. Possibly a program with three nested DO loops with a write statement inside all three writing out the three indices.

12. Given the formula $f = 6774./(2*(\ell + 3.3*r))$ where ℓ is the length of an organ pipe in inches, r is its radius in inches and f is its resonant frequency, determine f for $\ell \equiv 5''$, 15", 25", 35", and 45" and r = .5", 1.0", 1.5", 2.0", 2.5", using DO loops. Write out f, ℓ, and r with appropriate literal messages.

13. No. 10 of original assignment sheet.

14. Convert No. 7 to a subroutine.

15. Compute standard deviations.

16. Write a program to sort a list of numbers.

17. Write a subroutine using Newton's method for finding square roots. Compare to the predefined fcn SQRT.

18. Write a program to evaluate the truth value of logical expressions. (From a special lecture on logic by Mike Wasserman)

19. Numerical Integration-Simpson's rule.

20. Newton-Raphson method for finding roots of an equation.

EXAM NO. I, SECTION I-A

1. Place an I after those variables in the following list which are integer, an R after those which are real, and an X after those which are illegal.

JACK _____
BILL _____
3T89 _____
KOUNT _____

T4Z6 _____
INTEGER _____
MR. _____
ZEBRA _____

HOPE _____
NOW _____
4Z5 _____
LEMON _____

2. Translate the following expressions into FORTRAN.

$$\sqrt{\frac{u}{A} \cdot \frac{2A - R}{R}}$$ _____________________________

$$\left[\frac{B}{A} \cdot \frac{1 - e}{1 + e}\right]^2$$ _____________________________

$$\frac{s + A}{s + B^2 - 3C}$$ _____________________________

$$\frac{A^{(B + C)^{(D - A)}}}{B + C + \frac{A}{D + C}}$$ _____________________________

$$\frac{B^2 - A^2}{\sqrt{s}\,(s - A^{4/3})(s + A)}$$ _____________________________

$$\sqrt[3]{-\frac{B}{2} + \sqrt{\frac{B^2}{4} + \frac{A^3}{27}}}$$ _____________________________

3. Do the indicated operations.

$$\begin{array}{r} 111001_2 \\ +\ 10011_2 \\ \hline \end{array} \qquad \begin{array}{r} 110101_2 \\ -\ \ 1001_2 \\ \hline \end{array} \qquad \begin{array}{r} AB6D_{16} \\ +B413_{16} \\ \hline \end{array}$$

$$\begin{array}{r} E62A_{16} \\ -AC4D_{16} \\ \hline \end{array}$$

$11100101_2 = \underline{\qquad\qquad}_{16}$

$ABC_{16} = \underline{\qquad\qquad}_2$

$45_{10} = \underline{\qquad\qquad}_{16}$

$10010_2 = \underline{\qquad\qquad}_{10}$

$12_{10} = \underline{\qquad\qquad}_2$

4. Evaluate the following FORTRAN expressions.

If A = 3.5 and B = 2.0 then

A + B * (A − 1.) ** 2 * B/(A + B − .5) * (A − B) = ____________.

If A = 3.5, B = 2.0, and C = 4.2 then

A * B − C/B + A ** 2 = ____________.

If Y = 2 then

(Y − 1)/(Y ** 2 * (Y ** 2 + 5. * Y + 7)) = ____________.

If X = 3 then

(X ** 2 + 3)/(X * (X − 2.) * (X ** 2 + 2. ** + 4)) = ____________.

If A = 1.25, B = 3.5, C = 2.0, and X = 2.0 then

A * X ** 2 + B * X + C = ____________.

If A = 2.0, B = 4.0, C = 1.5 then

(−B + SQRT(B * B − 4. * A * C))/(2 * A) = ____________.

```
3 5 6 9 1 2 8 4 7 1 1 0 9 3 2 1 6 8 4 5 1 9 8 1 4 6 9 1 3
```

5. Using the above card representation, determine the values for the variables.

```
      READ(5,11)I, J, K, R, S, T
11    FORMAT(I3, I4, I2, 2F3.1, F6.2)
```

I=_______ J=_______ K=_______

R=_______ S=_______ T=_______

```
      READ(5,12)I, J, R, K, L, S, T
12    FORMAT(2(2I3, F4.2), F3.1)
```

I=_______ J=_______ R=_______

K=_______ L=_______ S=_______ T=_______

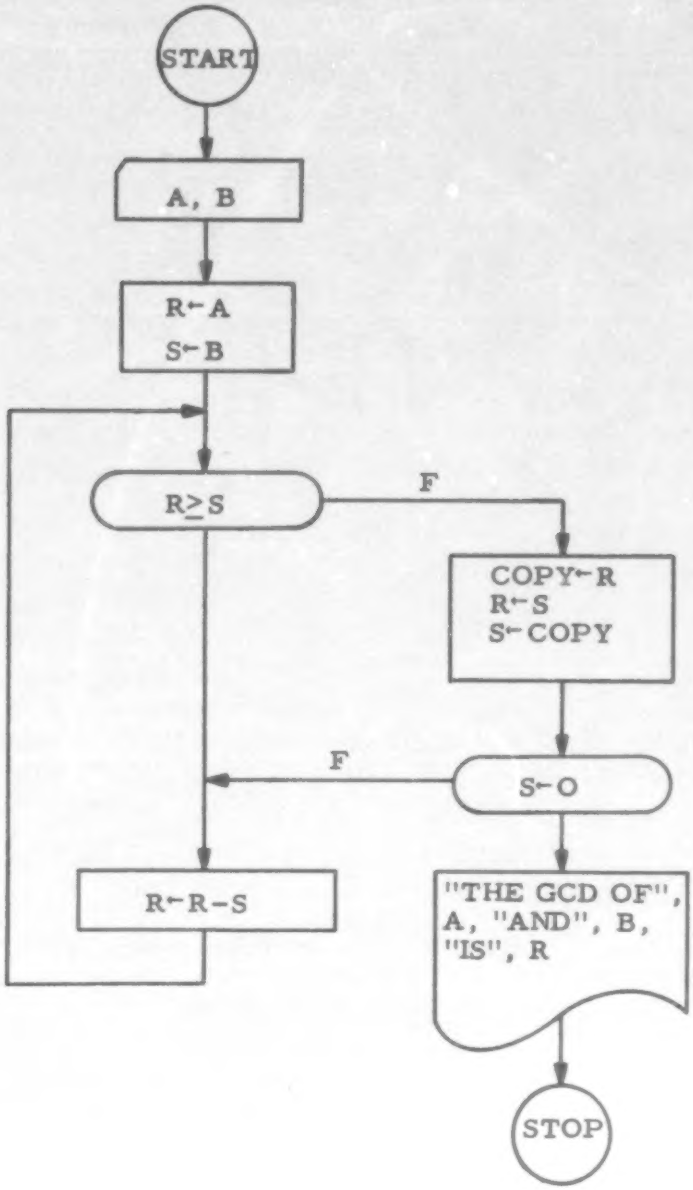
I=406, J=9125, K=82, L=110142, R=43.61
S=891.425, T=84.691, U=4562.91, V=4629.813

6. Using the above values write out the variables.

```
      WRITE(6,13)I, R, L, T, U, V
13    FORMAT(IX, I4, F6.1, I7, F5.2, F5.0, F6.1)
```

```
      WRITE(6,14)K, S, R, I, V, U, L, T, R
14    FORMAT(3(IX, I6, 2F6.1/))
```

7. **Write a program using the following flow chart:**


START
A, B
R←A
S←B
R≥S
F
COPY←R
R←S
S←COPY
F
S←O
R←R–S
"THE GCD OF", A, "AND", B, "IS", R
STOP

8. Assuming the values for A and B are 20.0 and 8.0 respectively, what is the value of R which is printed out.

R=____________

9. In the following program there are several Compiler mistakes. Below each write a small number in a circle and on the lines below identify the mistake. E.g., Mistake No. 1 is a misspelling of READ.

```
        REED(5,50)(B(I), I=1,4)
          ①
50      FORMAT(4F15.5

        READ(5,52)X

        VALUE=B(1)

        DO 10   R = 2,4

        VALUE = VALUE * X + B(R)

10      CONTINUE

        WRITE(6,51)VALUE

51      FORMAT(IX, F15.5)

        STOP
```

1	READ Misspelled	10	
2		11	
3		12	
4		13	
5		14	
6		15	
7		16	
8		17	
9		18	

EXAM NO. 1, SECTION I-B

(12 points)

I. Match the statements on the left with the <u>best</u> term on the right.

1. Read during execution time
2. Read during compilation time
3. Requires additional information
4. Result of finite word length
5. Result of insufficient field width
6. Signals end of compilation
7. Signals end of execution
8. Controls flow of jobs
9. Controls class (sometimes)
10. Name of real variable
11. Name of an integer variable
12. Illegal FORTRAN variable name

_____READ
_____STOP
_____******
_____Operating System
_____round-off error
_____your instructor
_____INSERT
_____QUIT
_____END
_____E16.8
_____program deck
_____data deck

(12 points)

II. a) Give a FORTRAN output statement for each of the following FORMAT statements. (8 points)

```
          |6|
1.      8 |  |FORMAT(6H1*10 =10X,E16.8//'A=',F15.8,6X,'AT',I10)
2.      9 |  |FORMAT(' NAME'T20'AGE'T35'SOCIAL SECURITY NUMBER')
3.     10 |  |FORMAT(1H 3(T6'I='4XI10,20XE15.8/))
4.     11 |  |FORMAT('0'T35'THE CRITICAL VALUES ARE'//4(10XE15.
          | 1|8, 15XE15.8/))
```

b) Give a FORMAT statement for the FORTRAN output statement given below. Skip to the 3rd line of a new page before printing. Include appropriate literals in your output. (4 points)

```
WRITE(6,5) LOOP,(A(I), B(I), I=1, 3)
```

(24 points)

III. Your instructor wrote the <u>program</u> given below at 3:00 a.m. in the morning after drinking three martinis. (Moral: Don't write a program at 3:00 a.m. after three martinis.) For each statement you are to (a) underline any error(s), (b) explain the error(s), and (c) write the corrected statement beneath it. If a statement has no errors, you must say so.

```
     6
THIS IS MY SUPER-NOTHING PROGRAM.

10    READ(4,5) NO, DATA

      I+1=SQRT(6./-NO*4.(DATA+1.)

      IF(A*I) 10, 10, 99

      DO20,NOTHIN=I+3, 6

20    IF((DATA+I)*2..EQ.3(-NOTHIN)) SUMTHN=DATA(NOTHIN)

 5    FORMAT(1H0,I10.4,E12.3)

99    END
```

(12 points)

IV. Write the following expressions in FORTRAN notation using as few parentheses as possible.

1. $A + \frac{B}{C+D}$ ______________________________

2. $(X_i + Y_\ell)X_\ell^2$ ______________________________

3. $\frac{(AX^2+BX + C)^{1/2}}{D}$ ______________________________

4. $(F_n - 3F_{n-1})(|F_{n+1}|)^{1/2}$ ______________________________

(40 points)

V. Write a program to first read a list of values for x, then compute y = f(x) for each value of x. Output the values of x and f(x) in a list with appropriate headings. Design your own formats. (a) flow chart (20 points), (b) code in FORTRAN (20 points).

$$y = \begin{cases} e^{x} & \text{if } x < 0.0 \\ 7.245x - 0.68x^{3} + 1 & \text{if } 0.0 \leq x < 11.6 \\ \dfrac{2.4x^{3/2} + 7.94}{\sqrt{|xe^{-x} - 1.7|}} & \text{if } 11.6 \leq x \leq 15.0 \\ 0.0 & \text{if } 15.0 < x \end{cases}$$

EXAM NO. 1, SECTION I-C

I. Write FORTRAN expressions for the following mathematical expressions: (2 points each).

1. $2\left(\frac{c-d}{e}\right)$

2. $\sqrt{a + b/2}$

3. $b^2 - yac$

4. $\frac{1}{2}\sin\left(-b + (c+d)^2\right)$

5. $(4+x)^y - \log_e(10.2)$

II. FORMAT PROBLEMS (2 points each)

A. Given the following format codes and numbers, punch the numbers on cards, without using decimal points.

6. F 8.3 ; 12.98

7. F 6.5 ; 8.72

B. Given the following format codes and cards, read the numbers as the computer would:

		1	2	3	4	5	6	7	8	9	10	11	12
8.	F10.4			2	7	0	3	9	7	8	0	2	
9.	F 3.1	3	0	9	2	7	0	0	4			7	2
10.	F 8.3						1						

III.

11. You are running the payroll program. Your last statement is WRITE (6,33) WAGE, RATE, TIME. Write a possible format statement 33 FORMAT .... Your statement should give format codes for WAGE, RATE, and TIME, as well as supplying appropriate literal messages. In addition, the printer should skip to a new page for each line of output. (10 points)

IV. The Computer is given the following program and data card. After the program is run, what are the final values of the variables?

PROGRAM:

```
      READ (6, 12) A,B,C
12    FORMAT (F6.2,F5.2,F6.2)
      I = 2 * A
      B = 10.1
      IF (A.LT.7.) GO TO 10
      J = 12
      I = I + 2
      D = A ** 4
      GO  TO  30
10    J = 10
      D = FLOAT (J)*A
      I = I+1
30    STOP
      END
```

DATA CARD	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
	6	.	3	8				2	.	1					3	.	2	9		

(2 points each)

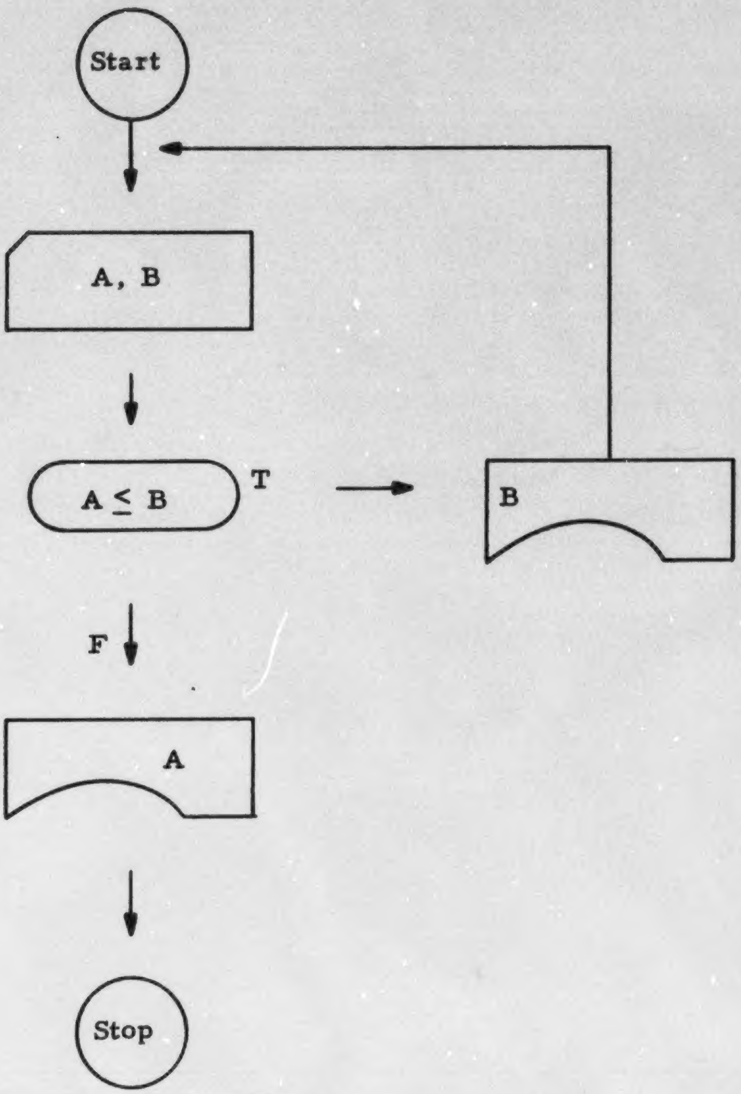
12. A =
13. B =
14. C =
15. D =
16. I =
17. J =

V. 18. The following program has four errors; find them. (5 points apiece)

The numbers on the right in parentheses are not statement numbers; they are line numbers and are to be used as reference numbers in answering.

```
        READ (5,100) (A(I),B(I),C(I),D(I),I=1,100)      (1)
        FORMAT (4F10.5)                                 (2)
        DO 10 J = 1,100                                 (3)
        E(J) = A(J) * B(J)                              (4)
        F(J) = C(J) * D(J)                              (5)
        WRITE (6,101) E(J), F(J)                        (6)
 101    FORMAT (2I10)                                   (7)
        GO TO (20,30,40,50)J                            (8)
  20    WRITE (6,102) J, A(J)                           (9)
        GO TO 10                                       (10)
  30    WRITE (6,102) J, B(J)                          (11)
        GO TO 10                                       (12)
  40    WRITE (6,102) J, C(J)                          (13)
        GO TO 10                                       (14)
  50    WRITE (6,102) J, D(J)                          (15)
        GO TO 10                                       (16)
  10    CONTINUE                                       (17)
 102    FORMAT (I10, F15.8)                            (18)
        STOP                                           (19)
        END                                            (20)
```

VI. 19. Translate the following flow chart into FORTRAN (18 points).

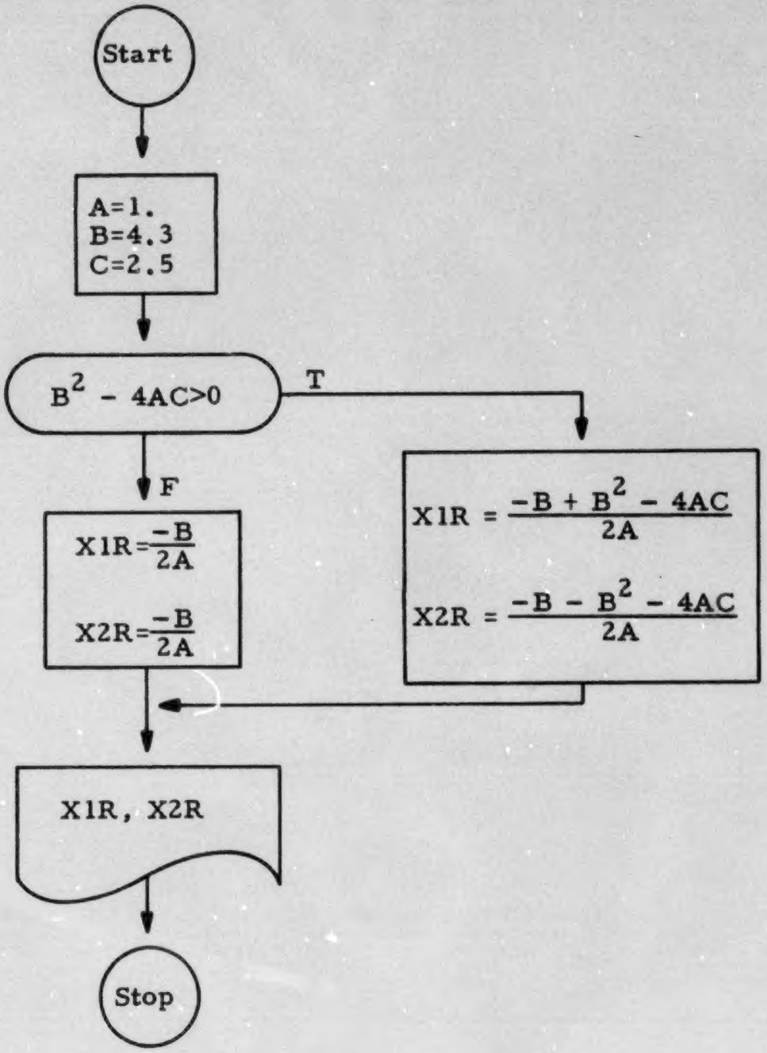


VII. 20. Devise a program to do the following: read 100 real numbers into the X array, find the smallest, and output it. (20 points)

EXAM NO. 2, SECTION I-B

(20 points)

I. **Write a FORTRAN program for the flow chart given below.**


Start
A=1.
B=4.3
C=2.5
$B^2 - 4AC > 0$
T
F
$X1R = \frac{-B}{2A}$
$X2R = \frac{-B}{2A}$
$X1R = \frac{-B + B^2 - 4AC}{2A}$
$X2R = \frac{-B - B^2 - 4AC}{2A}$
X1R, X2R
Stop

(20 points)

II. The computer was given the following program and data card. After the program was executed, what were the values of the variables I, K, A, and B?

```
    READ (5,301) A, B
    I=B
    A=10
    K=I+5
    B=2.*A-I
    K=I/K
301 FORMAT (F10.6,F10.8)
    STOP
    END
```

```
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
            7 6 4  8           1  0  .  4  8
```

I=______

K=______

A=______

B=______

EXAM NO. 1, SECTION II-E

(10 points)

1. Identify the following FORTRAN variable names as real, integer, or illegal.

		Real	Integer	Illegal
1.	A23X7			
2.	HAO6Q			
3.	MX2			
4.	I10.4			
5.	PRODUCT			
6.	Q*I			
7.	KAPPA			
8.	GTEEN			
9.	Q			
10.	LX4ZQB			

2. Write suitable FORMAT statements for the following READ and WRITE statements.

WRITE(6,1)A,Z24,N1,I,Q

WRITE(6,2)

READ(5,3) DAN,IS,A,NICE,GUY

READ(5,4) NEXT,LATEST,OLD,X

3. Write a FORMAT statement for the following WRITE statement so that the resulting computer printout appears as given below.

N=24,HI=31.65,RMID=41.24,RLOW=27.01,

AVG=18.673,VAR=64.938,DEV=8.316

WRITE(6,35)N,HI,RMID,RLOW,AVG,VAR,DEV

```
col 10                  40               70
2nd line g new page  TEST RESULTS FOR 24 STUDENTS
skip 2 lines
HI-RANGE=31.65 PERCENT,MID-RANGE=41.24 PERCENT,
LO-RANGE=27.01 PERCENT
AVERAGE=18.673, VARIANCE=64.938, STD.DEVIATION=8.316
```

EXAM NO. 2, SECTION II-E

(20 points)

1. Underline each error you find in the program below. Explain what the error is and write the correction beneath it.

```
    |6|
   M|Y|SUPER WHATZITDO PROGRAM.
  10| |READ(6,501)A,X
   5| |IF(X.EQ..7)10,20,5
    | |C + 2.*-A/B
  20| |GO TO 99
    | |IF(A-3.)B=SQRT(ABS(C)**1.75)
    | |FORMAT(F10.4,F8.16
    | |STOP
  99| |END
```

(20 points)

2. Write a program to read x, compute y as given below, and write the values of x and y. (a) flow chart; (b) code in FORTRAN

$$y = \begin{cases} 0 & \text{if } x < 0.0 \text{ or } x \geq 50. \\ -20.76 & \text{if } 0.0 \leq x < 10.9 \\ 16.19-x & \text{if } 10.9 \leq x < 21.6 \\ 3.00467x & \text{if } 21.6 \leq x < 50. \end{cases}$$

EXAM NO. 3, SECTION II-E

I. Write a FORTRAN arithmetic replacement statement for each of the following mathematical equations: (15 points)

1. $x = \frac{2y^2 - z}{y(3+x)}$

2. $v = \sum_{i=1}^{3} a_i x$

3. $d_\ell = \left(\sqrt{x_i^2 + y_i^2}\right) d_{k-1}^{i}$

4. $a_{ij} = b_{ik}\, c_{kj} + d_{il}$

5. $t_{n+1} = a_n \sin\left(r(3t_n)^{2/7}\right)$

II. Write a FORTRAN program to read in a 10 by 4 matrix (i.e., an array of 40 elements, 10 rows and 4 columns), find the smallest element and output its value and which element it is in the matrix, then output the complete matrix so that 4 columns of 10 elements each are printed. (a) flow chart; (b) code in FORTRAN. (25 points)

FINAL EXAM - FIRST SESSION

I. The following is a list of FORTRAN variable names. After those variables which are integer type, place an I, after real type place an R and place an X after any which are illegal. If it is illegal, circle the part which makes it wrong.

SPEED _____

IBM360 _____

LA _____

X-RAY _____

H2O _____

TIME _____

MODE _____

HELP _____

SCHNOOK _____

3RDANS _____

ANS2IS _____

II. Convert the following algebraic expressions to proper FORTRAN expressions.

$\frac{(S - A)(S + A)}{S^2 + A^2 + B^2}$ ____________________

$\frac{1}{S}\left(\frac{S - 1}{S}\right)^N$ ____________________

$\frac{1}{(S - A)^2 + B^2}$ ____________________

$\frac{(S - B)(S - C)}{S(S - A)}$ ____________________

$\frac{X^2 + 3}{(X - 2)(X^2 + 2X + 4)}$ ____________________

$\frac{X - 3}{(X - 2)^2(X^2 + X + 1)}$ ____________________

$(R + S)\sqrt{H^2 + (R - S)^2}$ ____________________

$$2PU\left(\frac{R}{2U - RV^2}\right)^{7/8}$$ ____________________________

$$\frac{R}{A}\sqrt{\frac{-A^2(1 - E^2) - 2AR + R^2}{1 - E^2}}$$

____

$$\frac{1}{R}\sqrt[3]{UAE^2\left(\frac{1 - (A - R)^2}{a^2E^2}\right)}$$

____

III. 1 6 4 3 9 8 1 1 0 4 8 2 6 5 1 2 7 8 3 9 1 4 6 2 8

Using the above card representation, fulfill the following READ statement:

```
      READ(5,11)I, R, IJ, T, Q, S
11    FORMAT(I6, F2.2, I3, 3F3.1)
```

I =________ R=________ IJ=________

T=________ Q=________ S=________

1 2 3 5 6.9 1 8 4 7.6 3 8 1 5.0 4 6 1 7 4 8 9.2 1 0 9 4.6

Using the above card representation, do as in the previous problem.

```
      READ(5,12)A, B, C, D, E, F
12    FORMAT(F7.2, F3.1, F5.2, F5.3, F4.2, F5.1)
```

IV. A=465.391, B=79.2468, C=5120.102, D=.983216,
I=4692, J=3, K=42, L=391, M=42156.

Using the above values, perform the following output statement:

```
      WRITE(6,13)A, I, B, J, C, D, K, L, M
13    FORMAT(IX, F5.1, I5, F5.2, I3, F5.2/F5.3, 3I4)
```

V. In the following write statement, how many numbers would be written out.

```
WRITE(6,14)((A(I,J),I=1,2),C(J),J=1,3)
```

VI. Locate the syntactical errors in the following program. Number each error found and on the lines below tell what the error is. For example, the first error is a misspelling of the READ statement.

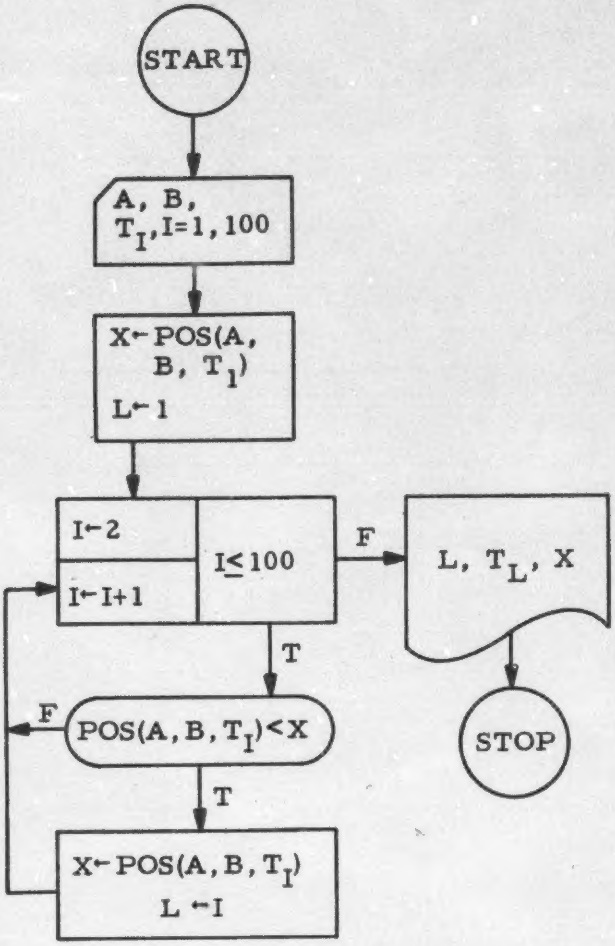
```
      REED(5,100)X, A
        ①
100   FORMAT(2F10.0)
      S=X
      T=X
      XS=S*T
      DO  4  I = 1,N
      IF(ABS(T).LE.A)GO  TO  5
      GO  TO  6
      WRITE(6,5) T(1)
  5   FORMAT(F10.6)
  6   RI = I
      T = -XS * T/62.*RI*(2. * RI + 1.)
      S = S + T
  4   CONTINUE
```

```
5   WRITE(6,9)X,S
    FORMAT(IX,'X=', 3X, F6.3, 3X, 'SIN(X)*', 3X, I10)
    STOP
```

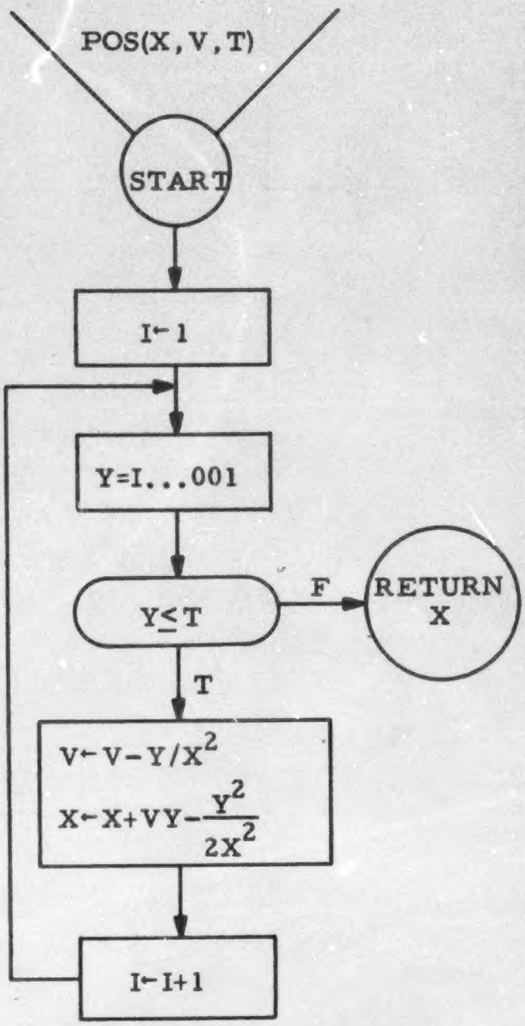
1	Misspelling of READ	9	
2		10	
3		11	
4		12	
5		13	
6		14	
7		15	
8		16	

VII. Following are flow charts for a program and a related function subprogram. Code the flow charts into FORTRAN.

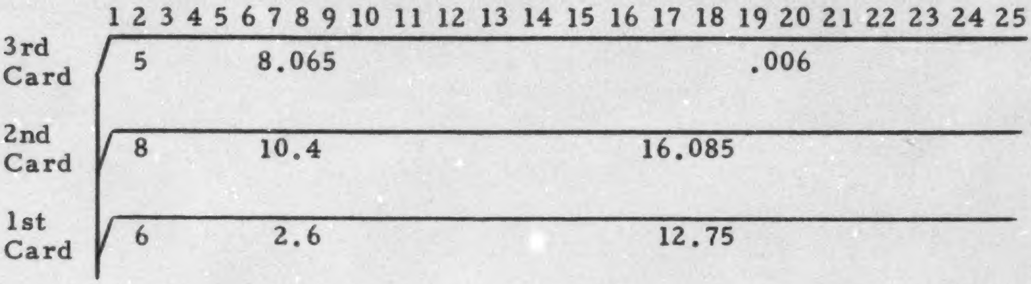
MAIN


START
A, B, T_I, I=1, 100
X← POS(A, B, T_1)
L← 1
I← 2
I← I+1
I≤ 100
F
L, T_L, X
STOP
T
F
POS(A, B, T_I) < X
T
X← POS(A, B, T_I)
L ← I

FUNCTION SUBPROGRAM


POS(X, V, T)
START
I← 1
Y=I...001
Y≤ T
F
RETURN X
T
V← V − Y/X^2
X← X+VY−$\frac{Y^2}{2X^2}$
I← I+1

VIII. The following program was run on a computer using the data cards given below. Trace the program and compute the values that were printed.


1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
3rd Card
5
8.065
.006
2nd Card
8
10.4
16.085
1st Card
6
2.6
12.75

```
      DIMENSION  X(2)
 20   READ(5,100) I, (X(K),K=1,2)
      K = 2 * I
      IF(K/5.LE.2)GO TO 20
      I = 3
      K = 2 + K/2 - I ** 2
 30   DO 80 L = K, I
      M = 8
      WRITE(6,101)L, K, I, X(1), X(2)
      GO TO (60, 61, 62)L
 60   M = 3.*ABS(X(L))/2.
      X(L) = M
      GO TO 80
 61   X(L) = -X(1)
      GO TO 80
 62   IF(X(2)/2.-M)80, 20, 30
 80   CONTINUE
100   FORMAT (I3, 2F10.4)
101   FORMAT (1HO, 3I10, 2F15.6)
      STOP
      END
```

L	K	I	X(1)	X(2)

IX. Let N be a positive integer. Recall that N! (read N factorial) is defined as N! = N(N−1)(N−2) ··· (3)(2)(1). Let K be a positive integer such that K < N. The binomial coefficient $\binom{N}{K}$ is defined as follows:

$$\binom{N}{K} = \frac{N!}{K!(N-K)!}$$

It is not hard to see that this number may also be expressed as

$$\binom{N}{K} = \frac{N(N-1)(N-2)\cdots(N-(K-1))}{K!}$$

Using either of these formulas, develop a flow chart and FORTRAN program to read two positive integers N,K (assume K < N), and compute and output the binomial coefficient $\binom{N}{K}$. Assume a format of 2I3 for your data cards; write your output according to format I10 (it is a theorem of mathematics that for all such N and K, $\binom{N}{K}$ is an integer). You may use a function subprogram if you wish. Partial credit will be given for either the flow chart or the program.

FINAL EXAM - SECOND SESSION

I. 1. Beside each of the following FORTRAN variable names write an R if it is type REAL, I if type INTEGER, or X if an illegal variable name.

HEX ________
LASTONE ________
NEXT ________
MID-PT ________
OK7D4 ________

SKIPIO ________
HELPME ________
2KEYS ________
IBM350 ________
KAREN ________

Beside each of the following FORTRAN constants write an R if it is type REAL, I if type INTEGER, or X if an illegal FORTRAN constant.

-0 ________
7E3 ________
5,341 ________
1.E ________
.79000D+03 ________

4.3E+0 ________
-933.333 ________
6I4 ________
21D.4 ________
+76300441 ________

2. This problem is on FORMAT statements and is in two parts:

PART 1: WRITE format
PART 2: READ format

Symbol Δ denotes a blank column anywhere it appears.

<u>PART 1</u>: Given a write statement

```
WRITE(6,100)MOD, QUE, RHO, TEE
```

for each of the following write a FORMAT statement that could produce such a line or lines with the given write statement.

1a) ΔΔ-1101ΔΔΔΔ0.45ΔΔΔΔ-196.2ΔΔΔΔ31.9632

1b) ΔΔM=-25ΔΔQ=2.ΔΔR=-12.3301ΔΔT=.92E-03

1c) Δ Δ Δ Δ Δ MOD = -100

Δ Δ Δ Δ ΔQUE = -3.2

Δ Δ Δ Δ ΔRHO = .9002

Δ Δ Δ Δ ΔTEE = 893110.66

1d) ΔMODΔCUEΔΔΔROWΔΔΔΔΔTEE

Δ-11Δ4.6ΔΔΔ-31.66ΔΔ.205

PART 2:

2a) A card is punched in the following FORMAT beginning in Column One:

ΔGOODΔLUCKΔONΔYOURΔPROJECT-SOULΔEXAMINATION

where symbol Δ denotes a blank column.

Assume a computer where one alphanumeric variable can hold 5 characters. Write a FORMAT statement to READ such a card.

2b) A card is punched in the following format:

Variable Name	Columns	Sample Format
NUM →	1-3	Δ 99
INT →	4-9	ΔΔ -101
TIME →	10-14	ΔΔ 33.
RATE →	15-22	ΔΔ 226.69
OFF →	23-30	ΔΔ -25.06

Write a format statement to READ such a card.

3. In each of the following you are to

   i - draw a flow chart of the decisions required

   ii - write statements to carry out the actions.

   These actions are a small part of a larger program, so you may assume that previous statements have given values to all variables and you need not write input and output statements.

   a. If $(\alpha + \beta) < 10^{-3}$ transfer to statement 120, otherwise transfer to statement 820

      1) using logical IF

      2) using arithmetic IF

   b. Place whichever of the variables Y and Z is algebraically larger in LARGE

      1) using logical IF

      2) using arithmetic IF

   c. If N=1,2, or 7, transfer to statement 122; if N=3,4 or 6 transfer to statement 123; if N=5,8 transfer to statement 124. STOP if it is not true that $1 \leq N \leq 8$.

   d. If $.555 \leq a \leq 1.555$, STOP, otherwise transfer to statement 622. Do this in two ways:

      1) with a logical IF having two relations combined with an .AND.

      2) with a logical IF having only one test, using the absolute value function.

4. Convert the following algebraic expressions to proper FORTRAN expressions.

$$\frac{(S - A)(S + A)}{S^2 + A^2 + B^2}$$

$$\frac{1}{S}\left(\frac{S-1}{S}\right)^{14}$$

$$\frac{X^2 + 3}{(X + 2)(X^2 + 2X + 4)}$$

$$2XY\left[\frac{R}{2X + 2Y^2}\right]^{7/6}$$

$$(R + S)\sqrt{H^2 + (R - S)^2}$$

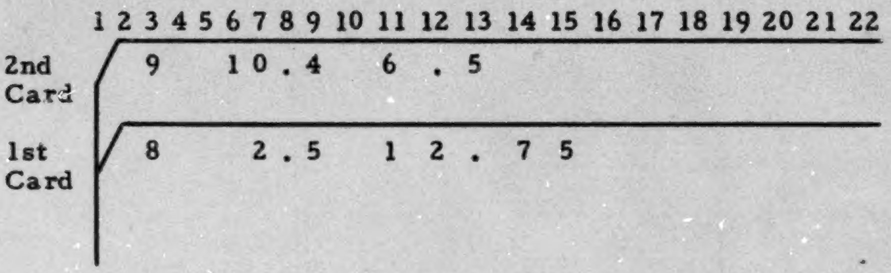
$$\frac{(S - B)(S - C)}{S(S + A)}$$

5. In the following program, locate the syntactical errors, place a number beside it and identify the error on the lines below. For example, the first errcr is a misspelling of the READ statement.

```
        REED(5,100)X,A
          ①
 100    FORMAT(2F10.0)
        S = X
        T = X
        XS = S * T
        DO  4  H = 1,N
        IF (ABS(T).LE.A)GO TO 5
        GO TO 6
        WRITE (6,5) T(1)
     5  FORMAT(F10.6)
     6  RI = H
        T = -XS * T/(2. * RI * (2. * RI + 1.)
        S = S + T
     4  CONTINUE
     5  WRITE (6,9) X,S
        FORMAT (IX, 'X = ', 3X, F6.3, 3X, 'S(X)' = , 3X, I10.2)
        STOP
```

1	Misspelling of READ	9	
2		10	
3		11	
4		12	
5		13	
6		14	
7		15	
8		16	

6. The following program was run on a computer using the data cards given below. Trace the program and fill in the table with the values that were printed.

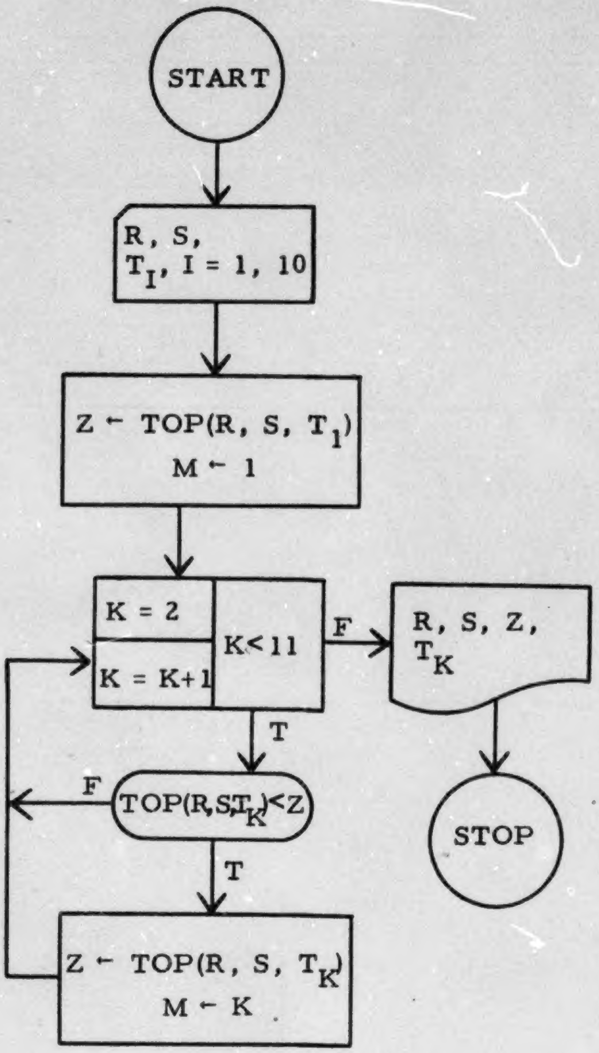

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22
2nd Card
9 10.4 6 . 5
1st Card
8 2.5 1 2 . 7 5

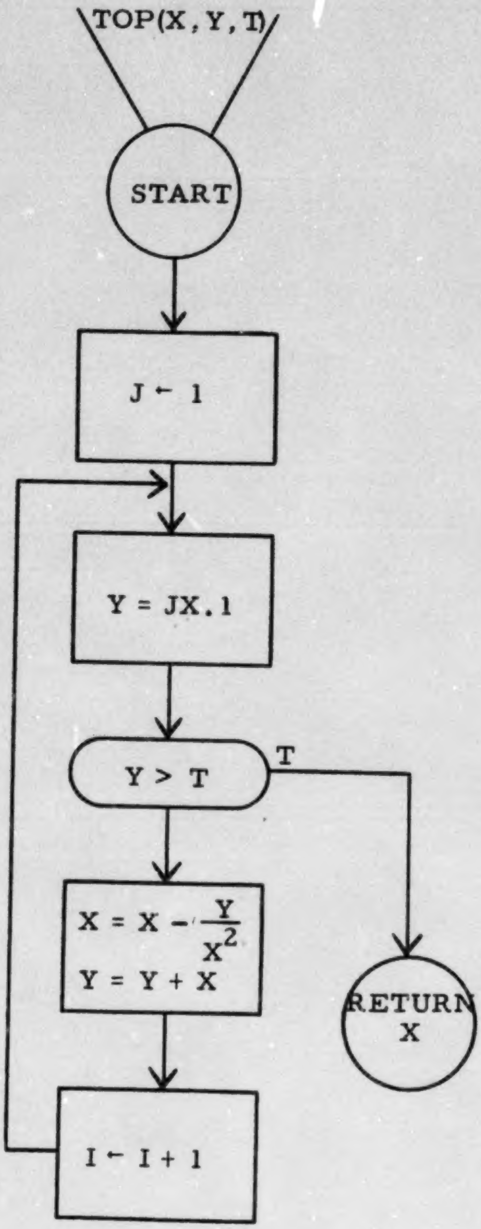
```
      DIMENSION A(2)
 20   READ(5,100)I, (A(K), K=1,2)
      M = 0
      K = 2 * (I-2)
      DO 80 L = I, K, 2
      N = K - L + 1
      WRITE(6,101)I, K, L, M, N, A(1), A(2)
      GO TO (20, 60, 60, 70, 61), N
 50   M = A(2)
      A(2) = 0.
      GO TO 80
 60   IF(A(1)-A(2))50, 90, 70
 61   M = A(1) + 1.
      A(1) = 1.
      GO TO 80
 70   M = M + A(1)
      A(1) = M/40
 80   CONTINUE
100   FORMAT(I3, 2F7.2)
101   FORMAT(1X, 5I10, 2F15.4)
 90   STOP
      END
```

VALUES PRINTED BY THE COMPUTER:

I	K	L	M	N	A(1)	A(2)

II. 1. Following are flow charts for a program and a related function routine. Code the flow charts into FORTRAN.


START
R, S,
T_I, I = 1, 10
Z ← TOP(R, S, T_1)
M ← 1
K = 2
K = K+1
K< 11
F
R, S, Z,
T_K
STOP
T
F
TOP(R,S,T_K)<Z
T
Z ← TOP(R, S, T_K)
M ← K


TOP(X, Y, T)
START
J ← 1
Y = JX.1
Y > T
T
X = X − $\frac{Y}{X^2}$
Y = Y + X
RETURN
X
I ← I + 1

2. Let N be a positive integer. Recall that N! (Read N Factorial) is defined as $N! = N(N-1)(N-2) \cdots (3)(2)(1)$. Let K be a positive integer such that $K \leq N$. The Binomial Coefficient is defined as

$$\binom{N}{K} = \frac{N!}{K!(N-K)!} \quad . \quad \text{(By definition } 0! = 1\text{)}$$

It is not hard to see that this number may also be expressed as

$$\binom{N}{K} = \frac{N(N-1)(N-2) \cdots (N-(K-1))}{K!} \quad .$$

Using either of the above formulas, develop a flow chart and FORTRAN program to read two positive real numbers p and q such that $p + q = 1.$ and a positive integer N and compute the quantity

$$S = \sum_{K=1}^{N} \binom{N}{K} p^{N} q^{N-K} \quad .$$

Assume an input format of 2F3.3 and I5. Write out p, q, N, and the value for the sum S. You may use a function routine if you wish. Partial credit will be given for either the flow chart or the program.

Good Luck!

APPENDIX G

Samples of Student Work

```
C      MATRICES
       REAL LARGE
       DIMENSION A (5,5)
       READ (5,10) ((A(I,J),J=1,5),I=1,5)
    10 FORMAT (5F10.5)
       WRITE (6,20) ((A(I,J),J=1,5),I=1,5)
    20 FORMAT (1X,'ORIGINAL MATRIX=',/,1X,5(F5.1,5X)/)
       LARGE=0
       DO 100 I=1,5
       DO 100 J=1,5
       IF (A(I,J).GT.LARGE) GO TO 105
       GO TO 100
   105 LARGE=A(I,J)
   100 CONTINUE
       SUM=A(1,1)+A(2,2)+A(3,3)+A(4,4)+A(5,5)
       WRITE (6,200) LARGE,SUM
   200 FORMAT (1X,'LARGEST=',1X,F10.2,/,1X,'SUM OF
      *PRINCIPAL DIAGONAL',1X,F10.2,//)
       WRITE (6,300)
   300 FORMAT (1X,'TRANSPOSE OF MATRIX=',/)
       M=0
    70 M=M+1
       IF (M.GT.5) GO TO 99
       WRITE (6,106) A(1,M),A(2,M),A(3,M),A(4,M),A(5,M)
   106 FORMAT (1X,5(F10.5,2X)/)
       GO TO 70
    99 STOP
       END
```

```
ORIGINAL MATRIX=
 42.2      95.5      67.7      23.0      65.5

12.0      16.0      18.5      23.0      57.6

31.0      37.0      73.0      81.5      44.5

15.0      25.5      52.0      96.7      63.2

55.0      45.5      73.2      34.2      66.5

LARGEST=      96.70
SUM OF                        PRINCIPAL DIAGONAL      294.40


TRANSPOSE OF MATRIX=

  42.20000    12.00000    31.00000    15.00000    55.00000

  95.50000    16.00000    37.00000    25.50000    45.50000

  67.70000    18.50000    73.00000    52.00000    73.20000

  23.00000    23.00000    81.50000    96.70000    34.20000

  65.50000    57.59999    44.50000    63.20000    66.50000

  STOP      0
```

```
C     PAYROLL PROGRAM
      SUM=0
      N=0
    3 READ (5,21) R1,R2,R3,RATE,TIME
      IF (RATE.EQ.0) GO TO 18
      N=N+1
      IF (TIME.GT.40) GO TO 9
      WAGE=RATE*TIME
    7 WRITE (6,22) R1,R2,R3,RATE,TIME,WAGE
      GO TO 15
    9 XTIME=TIME-40
      XRATE=1.5*RATE
      XWAGE=XRATE*XTIME
      RWAGE=RATE*40
      WAGE=RWAGE+XWAGE
      GO TO 7
   15 SUM=SUM+WAGE
      GO TO 3
   18 AVG=SUM/N
      WRITE (6,24) AVG
      STOP
   21 FORMAT (3A4,F4.2,F5.2)
   22 FORMAT (1X,3A4,2X,'RATE=',F4.2,2X,'TIME=',F5.2,2X,'WAGE=',F7.3)
   24 FORMAT (1X,'AVERAGE WAGE=',F7.3)
      END
```

```
AUL JONES     RATE=3.00  TIME=40.50  WAGE=122.250
HARLES OWEN   RATE=2.50  TIME=43.00  WAGE=111.250
IR BROWN      RATE=1.65  TIME=46.30  WAGE= 81.592
ENNIS JENG    RATE=1.25  TIME=45.00  WAGE= 59.375
ICTOR JEW     RATE=3.54  TIME=38.25  WAGE=135.405
OM SAWYER     RATE=5.00  TIME=40.00  WAGE=200.000
ARK TWAIN     RATE=2.85  TIME=48.50  WAGE=150.337
ALTED MILK    RATE=2.90  TIME=55.00  WAGE=181.250
ILLER DONO    RATE=4.15  TIME=35.75  WAGE=148.362
AYNE SIU      RATE=8.12  TIME=41.20  WAGE=339.416
```

```
C     GOING TO THE POST OFFICE
    1 READ (5,2) A,B,C
    2 FORMAT (3F8.2)
       IF (A.EQ.9991) GO TO 21
       IF (A-B) 3,3,4
    3 IF (B-C) 6,7,8
    5 IF (PD.LE.72) GO TO 9
      WRITE (6,10) A,B,C,PD
   10 FORMAT (1X,' A=',F8.2,5X,' B=',F8.2,5X,' C=',F8.2,5X,
     *'PARCEL DIMENSION =',F8.2,5X,'PACKAGE CAN''T BE MAILED')
      GO TO 1
    6 PD=C+2*(A+B)
       GO TO 5
    7 PD=A+2*(B+C)
       GO TO 5
    8 PD=B+2*(A+C)
       GO TO 5
    4 IF (A-C) 6,7,7
    9 WRITE (6,12) A,B,C,PD
   12 FORMAT (1X,' A=',F8.2,5X,' B=',F8.2,5X,' C=',F8.2,5X,
     *'PARCEL DIMENSION=',F8.2,5X,'PACKAGE CAN BE MAILED')
      GO TO 1
   21 STOP
      END
```

```
 A=    22.00     B=    24.00     C=    26.00     PARCEL DIMENSION =   118.00     PACKAGE CAN'T BE MAILED
 A=     5.00     B=     4.80     C=     6.00     PARCEL DIMENSION=    25.60     PACKAGE CAN BE MAILED
 A=    28.10     B=     5.00     C=     4.60     PARCEL DIMENSION=    47.30     PACKAGE CAN BE MAILED
 A=    41.00     B=    16.00     C=    32.12     PARCEL DIMENSION =   137.24     PACKAGE CAN'T BE MAILED
 A=     6.00     B=     9.20     C=     5.00     PARCEL DIMENSION=    31.20     PACKAGE CAN BE MAILED
 A=    18.50     B=    18.50     C=    18.50     PARCEL DIMENSION =    92.50     PACKAGE CAN'T BE MAILED
 A=    21.91     B=    21.90     C=    13.00     PARCEL DIMENSION =    91.71     PACKAGE CAN'T BE MAILED
 A=    18.00     B=    20.90     C=    22.10     PARCEL DIMENSION =    99.90     PACKAGE CAN'T BE MAILED
```

```
C     QUADRATIC EQUATIONS AND SOLUTIONS
      Q=0
    1 READ (5,100) A,B,C
      Q=Q+1
      IF (Q.EQ.8.) GO TO 10
      IF (A.EQ.0.AND.B.EQ.0.) GO TO 4
      IF (A.EQ.0.AND.B.NE.0.)GO TO 5
      R=B**2-4*A*C
      IF (R.EQ.0.) GO TO 6
      IF (R.LT.0.) GO TO 7
      X=(-B+SQRT(R))/(2*A)
      Y=(-B-SQRT(R))/(2*A)
      WRITE (6,104) A,B,C,X,Y
      GO TO 1
    4 WRITE (6,101) A,B,C
      GO TO 1
    5 X=-C/B
      WRITE (6, 102) A,B,C,X
      GO TO 1
    6 X=-B/(2*A)
      WRITE (6,102) A,B,C,X
      GO TO 1
    7 W=-B/(2*A)
      Z=SQRT(-R)/(2*A)
      WRITE (6,103) A,B,C,W,Z,W,Z
      GO TO 1
   10 STOP
  100 FORMAT (1X,3F10.2)
  101 FORMAT (1X,'A=', F10.5,5X,'B=',F10.5,5X,'C=',
     *F10.5,5X,'NO ROOTS EXIST')
  102 FORMAT (1X,'A=',F10.5,5X,'B=',F10.5,5X,'C=',
     *F10.5,5X,'X=',F15.7)
  103 FORMAT (1X, 'A=',F10.5,5X,'B=',F10.5,5X,'C=',
     *F10.5,/1X, 'REAL PART  OF X=',F15.7,5X,
     *'IMAGINARY PART OF X=',F15.7/1X,'REAL PART OF Y=',
     *F15.7,5X,'IMAGINARY PART OF Y=-', F15.7)
  104 FORMAT (1X, 'A=',F10.5,5X,'B=',F10.5,5X,'C=',
     *F10.5,5X,'X=',F15.7,5X,'Y=',F15.7)
      END
```

```
A=    2.10000       B=    2.10000       C=    2.10000
REAL  PART   OF  X=        -0.5000000      IMAGINARY  PART  OF  X=         0.8660250
REAL  PART  OF  Y=        -0.5000000     IMAGINARY  PART  OF  Y=-         0.8660250
A=    0.0           B=    5.00000       C=    4.00000       X=      -0.8000000
A=    0.0           B=    0.0           C=    8.50000       NO  ROOTS  EXIST
A=    8.00000       B=    2.00000       C=    9.00000
REAL  PART   OF  X=        -0.1250000      IMAGINARY  PART  OF  X=         1.0532684
REAL  PART  OF  Y=        -0.1250000     IMAGINARY  PART  OF  Y=-         1.0532684
A=    5.60000       B=   12.00000       C=    4.50000       X=      -0.4845828        Y=       -1.6582727
A=    1.00000       B=    1.00000       C=    1.00000
REAL  PART   OF  X=        -0.5000000      IMAGINARY  PART  OF  X=         0.8660250
REAL  PART  OF  Y=        -0.5000000     IMAGINARY  PART  OF  Y=-         0.8660250
A=    6.50000       B=    3.11000       C=    4.70000
REAL  PART   OF  X=        -0.2392307      IMAGINARY  PART  OF  X=         0.8159935
REAL  PART  OF  Y=        -0.2392307     IMAGINARY  PART  OF  Y=-         0.8159935
```

```
C      RESONANT FREQUENCIES FOR PIPES CHART
       DO 15 I=5,25,5
       DC 15 J=5,45,10
       R=I/10.
       L=J
       F=6774./(2*(L+3.3*R))
       WRITE (6,23) R,L,F
    23 FORMAT (1X,'RADIUS=',F3.1,10X,'LENGTH=',I2,30X,
      *'RESONANT FREQUENCY=',F8.3)
    15 CONTINUE
       WRITE (6,35)
    35 FORMAT (//,20X,'END OF CHART')
       STOP
       ENC
```

```
 RADIUS=0.5          LENGTH= 5                              RESONANT FREQUENCY= 509.323
 RADIUS=0.5          LENGTH=15                              RESONANT FREQUENCY= 203.423
 RADIUS=0.5          LENGTH=25                              RESONANT FREQUENCY= 127.092
 RADIUS=0.5          LENGTH=35                              RESONANT FREQUENCY=  92.415
 RADIUS=0.5          LENGTH=45                              RESONANT FREQUENCY=  72.605
 RADIUS=1.0          LENGTH= 5                              RESONANT FREQUENCY= 408.073
 RADIUS=1.0          LENGTH=15                              RESONANT FREQUENCY= 185.082
 RADIUS=1.0          LENGTH=25                              RESONANT FREQUENCY= 119.682
 RADIUS=1.0          LENGTH=35                              RESONANT FREQUENCY=  88.433
 RADIUS=1.0          LENGTH=45                              RESONANT FREQUENCY=  70.124
 RADIUS=1.5          LENGTH= 5                              RESONANT FREQUENCY= 340.402
 RADIUS=1.5          LENGTH=15                              RESONANT FREQUENCY= 169.774
 RADIUS=1.5          LENGTH=25                              RESONANT FREQUENCY= 113.088
 RADIUS=1.5          LENGTH=35                              RESONANT FREQUENCY=  84.781
 RADIUS=1.5          LENGTH=45                              RESONANT FREQUENCY=  67.808
 RADIUS=2.0          LENGTH= 5                              RESONANT FREQUENCY= 291.983
 RADIUS=2.0          LENGTH=15                              RESONANT FREQUENCY= 156.806
 RADIUS=2.0          LENGTH=25                              RESONANT FREQUENCY= 107.184
 RADIUS=2.0          LENGTH=35                              RESONANT FREQUENCY=  81.418
 RADIUS=2.0          LENGTH=45                              RESONANT FREQUENCY=  65.640
 RADIUS=2.5          LENGTH= 5                              RESONANT FREQUENCY= 255.623
 RADIUS=2.5          LENGTH=15                              RESONANT FREQUENCY= 145.678
 RADIUS=2.5          LENGTH=25                              RESONANT FREQUENCY= 101.865
 RADIUS=2.5          LENGTH=35                              RESONANT FREQUENCY=  78.312
 RADIUS=2.5          LENGTH=45                              RESONANT FREQUENCY=  63.606


                    END CF CHART
 STOP      0
```

APPENDIX H

Evaluation Questionnaire

I. The following questions relate to your general impressions. (Save specific comments regarding course content for later.)

a) What were the good features of the program?

b) What were the bad features of the program?

c) Generally, how pleased were you with the program?

d) Where were you displeased or dissatisfied?

e) If the course were to be taught again, what changes in the objectives, scope, or whatever, would you recommend?

II. The following questions relate to the course content?

f) What were the good features of the course outline? Was any part of it found to be especially good?

g) What were the bad features? What material presented a lot of difficulty?

h) What would you change? If you did not follow the outline, give your version.

III. If you have any other comments on any phase of Project Soul which you would like to see included in the Final Report, give them.